UNIVERSITÀ DEGLI STUDI DI BERGAMO | Dipartimento di Ingegneria Gestionale, dell'Informazione e della Produzione

ADAPTIVE LEARNING, ESTIMATION AND SUPERVISION OF DYNAMICAL SYSTEMS (ALES)

Glossary

Master Degree in COMPUTER ENGINEERING

Data Science and Data Engineering Curriculum

SPEAKER
Prof. Mirko Mazzoleni

PLACE
University of Bergamo

Glossary

General

- x: scalar
- $\boldsymbol{x}$: vector
- X: matrix
- $\mathbf{X}$: matrix

Dynamical systems

- $\boldsymbol{\varphi}(t)$: regressors vector
- $\boldsymbol{z}(t)$: instrumental variables vector
- $A(z), B(z), C(z), D(z), F(z)$: transfer function polynomials
- $G(z)$: input-output transfer function
- A, B, C, D: state-space matrices
- $u(t), y(t)$: scalar input/output
- $\boldsymbol{u}(t), \boldsymbol{y}(t)$: multivariable input/output (vector)
- U, Y: stacking of scalar inputs/outputs
- $\mathbf{U}, \mathbf{Y}$: stacking of multivariable inputs/outputs vectors

UNIVERSITÀ DEGLI STUDI DI BERGAMO | Dipartimento di Ingegneria Gestionale, dell'Informazione e della Produzione

ADAPTIVE LEARNING, ESTIMATION AND SUPERVISION OF DYNAMICAL SYSTEMS (ALES)

Lecture 1: Introduction

Master Degree in COMPUTER ENGINEERING

Data Science and Data Engineering Curriculum

SPEAKER
Prof. Mirko Mazzoleni

PLACE
University of Bergamo

Who I am

- **Name:** Mirko Mazzoleni
- **Currently:** Assistant Professor (RTD-B), **Control and Automation Lab (CAL UniBG)**
  - ✓ *Research:* System identification, machine learning, fault diagnosis
  - ✓ *Teaching:*
    1. Identificazione dei Modelli e Analisi dei Dati (IMAD)
    2. Adaptive learning, estimation and supervision of dynamical systems (ALES)
    3. Gestione, analisi e rappresentazione dei dati (GARD)
- **Other:** Co-founder of AISent srl startup https://aisent.io/ AISENT
- **Contacts**
  - ✓ mirko.mazzoleni@unibg.it
  - ✓ https://mirkomazzoleni.github.io/
  - ✓ http://cal.unibg.it/ **Website CAL UniBG**
  - ✓ https://www.facebook.com/ControlAutomationLabUnibg/

Outline

1. Presentation of the Adaptive Learning, Estimation and Supervision of dynamical systems (ALES) course
2. What we have learnt so far…
3. …and what we still have to learn!

Outline

1. **Presentation of the Adaptive Learning, Estimation and Supervision of dynamical systems (ALES) course**
2. What we have learnt so far…
3. …and what we still have to learn!

Prerequisites for the course

It is **strongly suggested to have** good foundations in the following topics:

- Systems identification (IMAD)
- Control systems
- Linear algebra
- Calculus 1 and 2

How to «refresh» the prerequisites?

- Systems identification (IMAD) ⇨ Unibg course
- Control systems ⇨ Unibg courses
- Linear algebra ⇨ Unibg course, Gilbert Strang course @MIT on Youtube
- Calculus I and II ⇨ UniBg course

Exam

- Short **written** exam **1 hour** on the theory **+ coding project** with oral discussion

Open questions about the theory

Matlab implementation of a technique\algorithm from the literature + oral power-point presentation

How to be prepared for the exam?

- Follow the lessons
- Study the theory
- Implement by yourself the algorithms presented at lesson

Educational objectives of the course

At the end of the course, you will be able to:

- **Employ** *recursive* and *adaptive estimation methods*
- **Identify** models of dynamical systems in *closed-loop* settings and in *state-space form*, with *multiple-inputs and multiple-outputs* (MIMO)
- **Define** and **solve** industrial *fault diagnosis problems*, identify their main components and envisage a possible solution using different approaches

What to expect?

	2021/2022		
	Mean	N	% Voti < 6
Was the preliminary knowledge possessed sufficient to understand the topics included in the exam programme?	8,2	5	0
Is the study load of the course proportionate to the credits assigned?	6,4	5	60
Is the teaching material (indicated and available) adequate for studying the subject?	9,4	5	0
Have the examination methods been clearly defined?	8,2	5	0
Are the times for lessons, exercises and any other educational activities respected?	10	5	0
Does the teacher stimulate/motivate interest in the discipline?	9,4	5	0
Does the teacher explain the topics clearly?	10,00	5	0
Supplementary teaching activities such as exercises, tutoring, workshops, etc. (language training is not included) where existing, are they useful for learning the subject?	8,8	5	0
Was the teaching carried out in a manner consistent with what was stated on the study course website?	9,4	5	0
Is the teacher available for clarifications and explanations?	10,00	5	0
Are you interested in the topics covered in the course?	7,2	5	20

Teaching materials

Materials provided by the teacher

- Lectures slides
- **Sorry. No code** ☹: you have to implement the algorithms by your own MATLAB

Interaction and feedback

- During the week I will give you **activities** to do and **tests** to answer. They are **optional** but they help you to understand the degree of learning before the exam. In addition, they will contribute to giving a bonus of **+3 points** to the final grade

We will use **MS Teams activities**

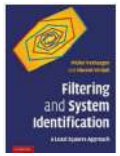
Teaching materials

Suggested textbooks

- Peter C. Young, **Recursive Estimation and Time-Series Analysis**, 2° ed. Springer (2011)

- Rolf Isermann, **Fault-Diagnosis Systems: An Introduction From Fault Detection To Fault Tolerance**, Springer (2005)

- Michel Verhaegen, **Filtering and system identification: a least squares approach**, Cambridge University Press (2007)

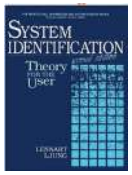


- Jie Chen, Ron J. Patton, **Robust Model-Based Fault Diagnosis for Dynamic Systems**, Springer (1999)

Teaching materials

Other suggested textbooks

- Lennart Ljung, **System Identification: Theory for the User**, Pearson (1998)



- Torsten Soderstrom, Petre Stoica, **System identification** Prentice Hall international (2001)

- Peter Van Overschee, Bart De Moor, **Subspace identification for linear systems**, Springer (1996)

- Janos Gertler, **Fault Detection and Diagnosis in Engineering Systems**, CRC Press, (1998)

- Steven X. Ding, **Model-baed fault diagnosis techniques**, 2° ed, Springer (2013)

Syllabus

1. **Recursive and adaptive identification**
   - 1.1 Recursive ARX estimation (RLS)
   - 1.2 Least Mean Squares (LMS)
   - 1.3 Instrumental Variables (IV)
2. **Closed-loop identification**
3. **Subspace and MIMO identification**
   - 3.1 Singular Value Decomposition
   - 3.2 Impulse data: Ho-Kalman, Kung algorithms
   - 3.3 Generic I/O data: the MOESP algorithm
4. **Supervision of dynamical systems**
   - 4.1 Introduction to fault diagnosis
   - 4.2 Model-based fault diagnosis
   - 4.3 Parity space approaches
   - 4.4 Observer-based approaches
   - 4.5 Signal-based fault diagnosis
   - 4.6 Knowledge-based fault diagnosis

CASE STUDIES

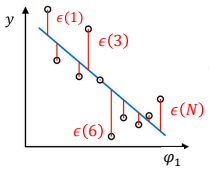
- Virtual Reference Feedback Tuning
- Nuclear particles classification
- Leak detection in an industrial valve
- Bearing fault identification

Outline

1. Presentation of the Adaptive Learning, Estimation and Supervision of dynamical systems (ALES) course
2. **What we have learnt so far…**
3. …and what we still have to learn!

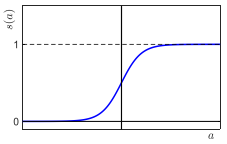
Modeling static systems and machine learning

Linear regression model



$$y(i) = \boldsymbol{\varphi}^{\mathsf{T}}(i)\boldsymbol{\theta} + \epsilon(i)$$

Logistic regression model



$$P(y = 1|\boldsymbol{\varphi}) = \frac{1}{1 + e^{-\boldsymbol{\varphi}^{\mathsf{T}}\boldsymbol{\theta}}}$$

Linear regression algorithm

$$J(\boldsymbol{\theta}) = \frac{1}{N}(X\boldsymbol{\theta} - Y)^{\mathsf{T}}(X\boldsymbol{\theta} - Y)$$

$$\widehat{\boldsymbol{\theta}} = (X^{\mathsf{T}}X)^{-1}X^{\mathsf{T}}Y$$

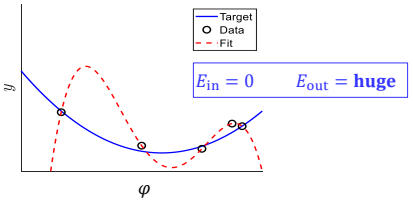
Logistic regression algorithm

$$J(\boldsymbol{\theta}) = -\sum_{i=1}^{N}\left(y(i)\ln\pi(i) + (1 - y(i))\ln[1 - \pi(i)] \right)$$

Gradient descent $\widehat{\boldsymbol{\theta}}^{(k+1)} = \widehat{\boldsymbol{\theta}}^{(k)} - \alpha \cdot \nabla J(\boldsymbol{\theta})\Big|_{\boldsymbol{\theta}=\widehat{\boldsymbol{\theta}}^{(k)}}$

Modeling static systems and machine learning

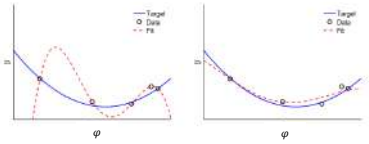
Overfitting



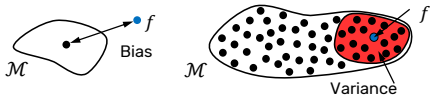
Regularization

$$E_{\text{aug}}(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N}\left(y(i) - h(\boldsymbol{\varphi}(i);\,\boldsymbol{\theta})\right)^2 + \lambda_{\text{reg}} \cdot \sum_{j=0}^{d-1}(\theta_j)^2$$

λ_{reg_1} $\lambda_{\text{reg}_2} > \lambda_{\text{reg}_1}$



Bias and variance



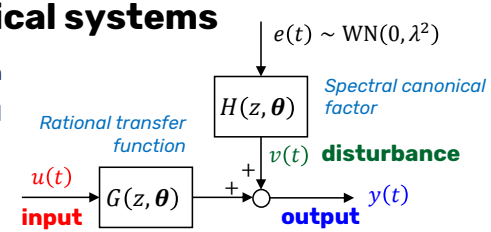
Validation and cross-validation

Train (60%)	Validation (20%)	Test (20%)

Modeling LTI SISO dynamical systems

In the IMAD course, we studied **Prediction Error Methods** (PEM) for **SISO LTI** dynamical systems

The **general form** of the models was:



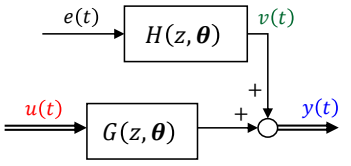
- $e(t)$ is **a white noise** signal, introduced to account for all **unmodeled dynamics** and the effects of **external unmeasured disturbances**
- $G(z;\boldsymbol{\theta})$ and $H(z;\boldsymbol{\theta})$ are suitable **transfer functions** that depend on parameters $\boldsymbol{\theta}$

$$\mathcal{M}(\boldsymbol{\theta}): y(t) = G(z, \boldsymbol{\theta})u(t-1) + H(z, \boldsymbol{\theta})e(t), \qquad e(t) \sim \text{WN}(0, \lambda^2)$$

Modeling LTI SISO dynamical systems

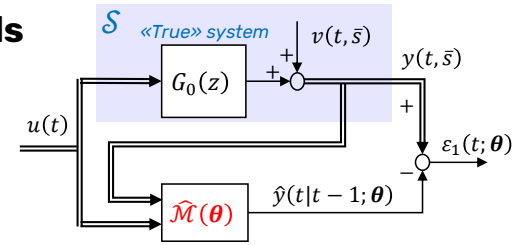
Family of models	$G(z, \boldsymbol{\theta})$	$H(z, \boldsymbol{\theta})$
ARX	$\frac{B(z,\boldsymbol{\theta})}{A(z,\boldsymbol{\theta})}z^{-k}$	$\frac{1}{A(z,\boldsymbol{\theta})}$
ARMAX	$\frac{B(z,\boldsymbol{\theta})}{A(z,\boldsymbol{\theta})}z^{-k}$	$\frac{C(z,\boldsymbol{\theta})}{A(z,\boldsymbol{\theta})}$
OE	$\frac{B(z,\boldsymbol{\theta})}{F(z,\boldsymbol{\theta})}z^{-k}$	1
FIR	$B(z,\boldsymbol{\theta})z^{-k}$	1
BJ	$\frac{B(z,\boldsymbol{\theta})}{F(z,\boldsymbol{\theta})}z^{-k}$	$\frac{C(z,\boldsymbol{\theta})}{D(z,\boldsymbol{\theta})}$

The $\boldsymbol{\theta}$ vector represents the **model parameters** (value of the coefficients of the $A(z), B(z), C(z)$ polynomials)



Prediction Error Methods

Given a model family $\mathcal{M}(\boldsymbol{\theta})$, the estimate of $\boldsymbol{\theta}$ was performed by minimizing the variance of the **one-step prediction error** $\varepsilon_1(t;\boldsymbol{\theta})$



Model in «prediction form»

Model in prediction form

$$\hat{\mathcal{M}}(\boldsymbol{\theta}):\ \hat{y}(t|t-1;\boldsymbol{\theta}) = H^{-1}(z,\boldsymbol{\theta})G(z,\boldsymbol{\theta})\cdot u(t) + [1 - H^{-1}(z,\boldsymbol{\theta})]\cdot y(t)$$

PEM estimate

$$\varepsilon_1(t;\boldsymbol{\theta}) = y(t) - \hat{y}(t|t-1;\boldsymbol{\theta})$$

$$J_N(\boldsymbol{\theta}) = \frac{1}{N}\sum_{t=1}^{N}\varepsilon_1(t;\boldsymbol{\theta})^2 \qquad \hat{\boldsymbol{\theta}}_N = \arg\min_{\boldsymbol{\theta}\in\Theta} J_N(\boldsymbol{\theta})$$

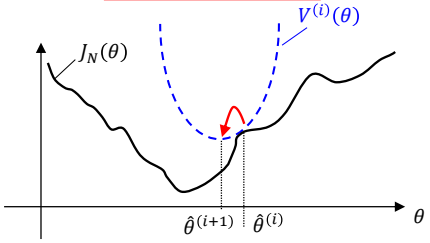
Prediction Error Methods: estimation algorithms

- **ARX, FIR: convex** cost function (global minimum) ⇨ **Least squares**
- **ARMAX, BJ, OE: not convex** cost function (local minima) ⇨ **Quasi-Newton iterative schemes**

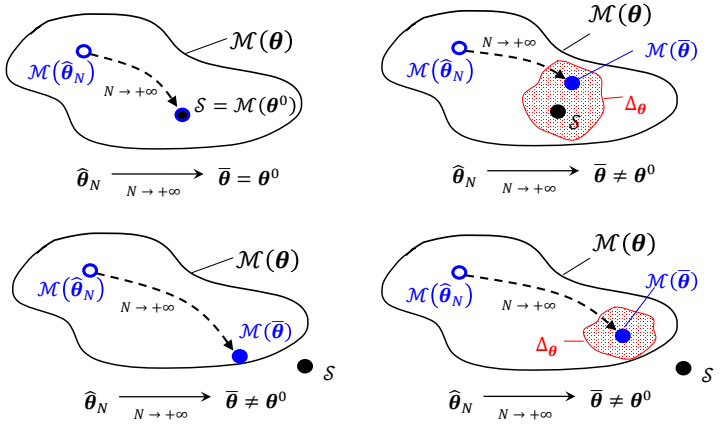
ARX, FIR

$$\hat{\boldsymbol{\theta}}_N = \left[\sum_{t=1}^{N}\boldsymbol{\varphi}(t)\boldsymbol{\varphi}^{\mathsf{T}}(t)\right]^{-1}\cdot\left[\sum_{t=1}^{N}\boldsymbol{\varphi}(t)y(t)\right]$$

ARMAX, BJ, OE

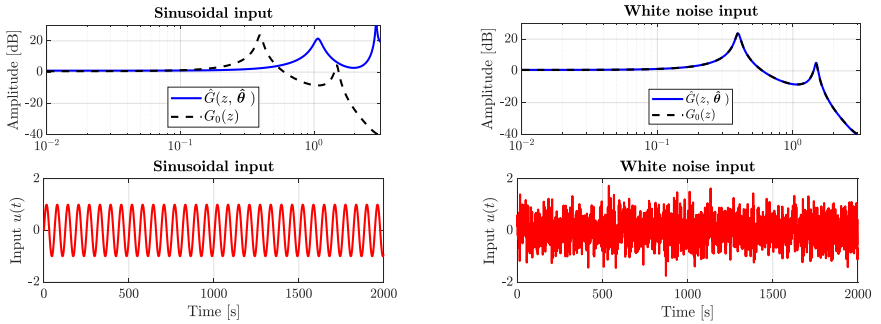


Prediction Error Methods: asymptotic properties



Prediction Error Methods: persistent excitation

Experimental identifiability requires the input signal to be **persistently exciting** of enough order, as **white noise, Pseudo-Random Binary Signal (PRBS), Multisine**

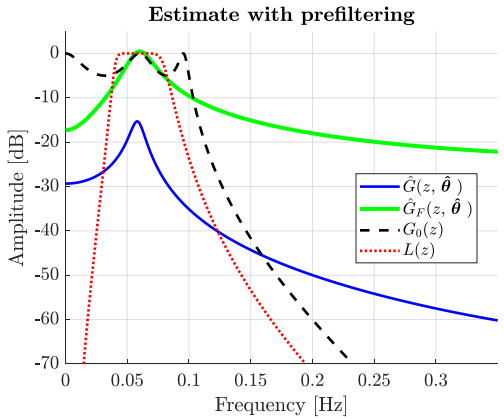


Prediction Error Methods: robustness

When the chosen model is not able to exactly represent the true system, we can **prefilter the data** to perform **bias-shaping**

Focus the modelling efforts in a selected bandwidth

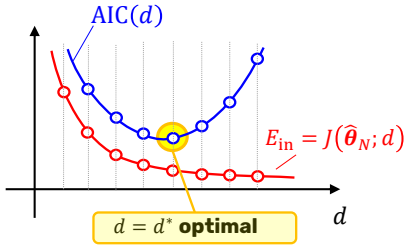
$$\bar{J}(\boldsymbol{\theta}) = \frac{1}{2\pi}\int_{-\pi}^{\pi} \left|\Delta G\left(e^{j\omega}, \boldsymbol{\theta}\right)\right|^2 \cdot g_F(\omega, \boldsymbol{\theta})\, d\omega$$



Prediction Error Methods: validation techniques

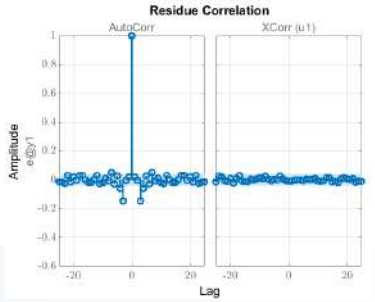
Choosing the model complexity

- Complexity formulas
- Validation



Assessing model goodness

- Residual analysis
- Uncertainty of the estimates
- Prediction and simulation fitness



Empirical Transfer Function Estimate (ETFE)

Suppose we divide the signals $u(t)$ and $y(t)$, both multisines, into sub-sequences $u^{[p]}, y^{[p]}$ which contain the different $p = 1, \ldots, P$. We define:

$\breve{U}^{[p]} = \mathrm{DFT}\left(u^{[p]}\right)$ **input DFT**

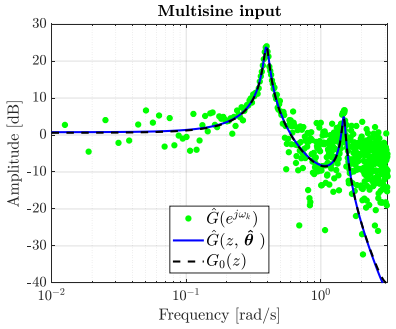
$\breve{Y}^{[p]} = \mathrm{DFT}\left(y^{[p]}\right)$ **output DFT**

$$\hat{U}(k) = \frac{1}{P}\sum_{p=1}^{P} \breve{U}^{[p]}(k)$$

$$\hat{Y}(k) = \frac{1}{P}\sum_{p=1}^{P} \breve{U}^{[p]}(k)$$

$$\Rightarrow \quad \hat{G}(k) = \frac{\hat{Y}(k)}{\hat{U}(k)}$$

Empirical Transfer Function Estimate (ETFE)



Outline

1. Presentation of the Adaptive Learning, Estimation and Supervision of dynamical systems (ALES) course
2. What we have learnt so far…
3. **…and what we still have to learn!**

Adaptive Learning, Estimation and Supervision of dynamical systems

A) ADAPTIVE LEARNING

The identification algorithms we have learnt so far process all available data as a **single batch.** This approach has several drawbacks:

- **memory occupancy:** we need to store all data in the computing device
- **computational load:** we may need to perform a big matrix inversion
- **time delay:** if we need to take quick decisions, an immediately available - albeit approximate – model, may be more useful than a perfect model available only after a batch of data has been collected

Adaptive Learning, Estimation and Supervision of dynamical systems

In many applications data are not available all together, but they are **collected one by one on-line**. Can we still identify a model in these conditions?

RECURSIVE IDENTIFICATION METHODS

Sometimes the system generating the data is **not time-invariant** (or it is nonlinear).

Can we recompute the model **on-line** adapting it to the current behavior of the system?

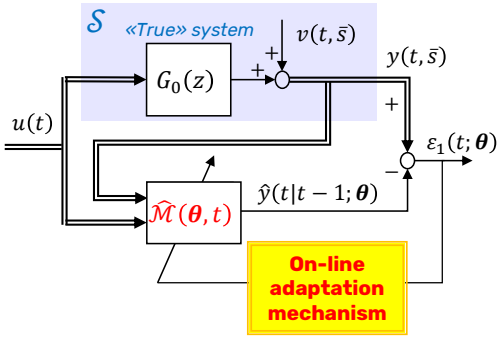
ADAPTIVE IDENTIFICATION METHODS

Adaptive Learning, Estimation and Supervision of dynamical systems

When a model is:

- time-varying, or
- Nonlinear but linearly approximated

the linear model has only **local validity** and must be **online updated** to the most recent data available *(model tracking)*



Adaptive Learning, Estimation and Supervision of dynamical systems

The main applications of **recursive and adaptive learning methods** are:

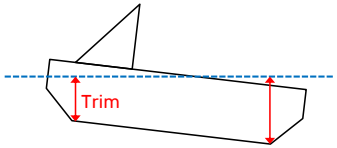
1. **Adaptive filtering and control**: a time-varying controller or filter is tuned to the **currently estimated** model of the process *(iterative learning control, extremum-seeking control, model reference adaptive control, adaptive filtering and equalization)*

2. **Fault diagnosis**: a model of the system is continuously updated to rapidly identify significant **changes in its parameters**. For LTI systems, a change in a parameter might be a symptom of a system degradation due to a fault *(diagnosis of parametric faults)*

Example: ship steering adaptive control

A **ship's heading angle** and position is controlled using the **rudder angle**. For large ships, its response to a change in rudder angle is so **slow** that it is affected by random components as **wind** and **wave motion**

The steering dynamics of the ship depends on its shape, size, loading, trim, and water depth. In particular, **loading** and **water depth** may **vary** with time



The rudder angle controller must be **continuously retuned** to match the dynamics of the system

Example: self-tuning predictive control

Let the system to be controlled an $\mathrm{ARX}(n_a = 1, n_b = 2, k = 1)$ so that:

$$y(t) = ay(t-1) + b_0 u(t-1) + b_1 u(t-2) + e(t) \qquad e(t) \sim \mathrm{WN}(0, \lambda^2)$$

Under the standard hypotheses for **Minimum Variance (MV) control**, the MV control law for a reference signal $y^0(t)$ is:

Controlli automatici - 6 CFU

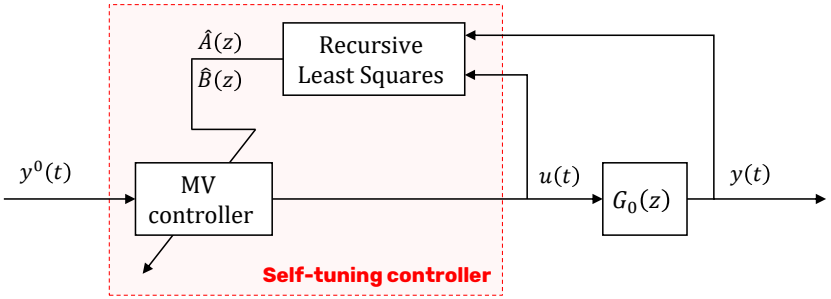
$$u(t) = \frac{1}{b_0}\left(-b_1 u(t-1) - ay(t-1) + y^0(t)\right)$$

If the system changes its parameters in time, this control law will be **no more «optimal»** in terms of the **tracking error variance**

$$\mathbb{E}\left[\left(\hat{y}(t|t-1) - y^0(t)\right)^2\right]$$

Example: self-tuning predictive control

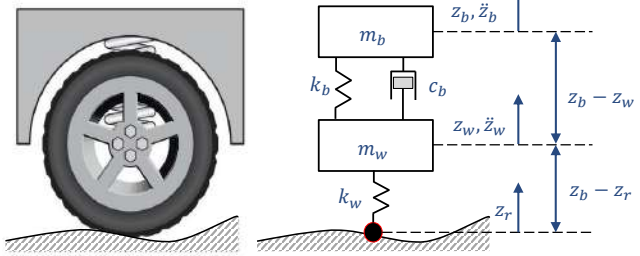
Thus, an **adaptive (self-tuning) control scheme** is envisaged:



The estimation algorithm will provide the estimates $\hat{a}(t), \hat{b}_0(t), \hat{b}_1(t)$ at **each time** t

Example: supervision of an automotive suspension

The **suspension** and the **tire pressure** have a large influence on the vehicle dynamics and are thus highly **safety critical**



Consider a (simplified) **quarter-car** model, where:

- m_b, m_w: body and wheel mass
- $z_b - z_w$: suspension deflection
- k_w: wheel rigidity
- k_b, c_w: suspension rigidity and damping coefficients
- z_r: road level

Example: supervision of an automotive suspension

Force balances lead to:

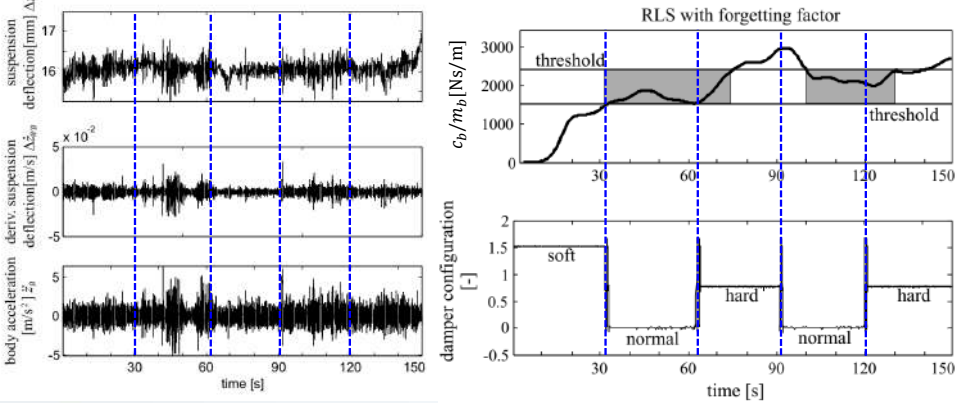
$$\begin{cases} m_b \ddot{z}_b(t) = -k_b\big(z_b(t) - z_w(t)\big) - c_b\big(\dot{z}_b(t) - \dot{z}_w(t)\big) - m_b g \\ m_w \ddot{z}_w(t) = k_b\big(z_b(t) - z_w(t)\big) + c_b\big(\dot{z}_b(t) - \dot{z}_w(t)\big) - k_w\big(z_w(t) - z_r(t)\big) - m_w g \end{cases}$$

Assume that the body mass m_b and damper spring rigidity k_b are known. It is possible to **estimate the suspension viscous coefficient** c_b with **linear regression** measuring the body acceleration $\ddot{z}_b$ and the suspension deflection $z_b - z_w$

However, for a **fault detection** purpose (e.g. a **worn out of the suspension** which cause a change in the damping coefficient) **recursive methods** are needed

Example: supervision of an automotive suspension

Variation of damper configuration at $t \approx 30, 60, 90, 120$ s



Adaptive Learning, Estimation and Supervision of dynamical systems

B) ESTIMATION AND IDENTIFICATION

Until now, we focused on **Prediction Error Methods (PEM)** for **SISO** LTI system identification. These methods are optimal if some assumptions holds:

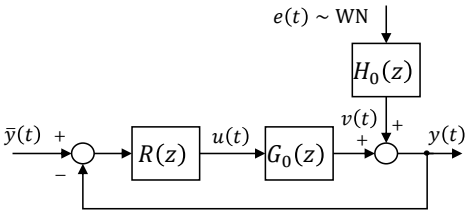
- The input and output signals are assumed to be **stationary** stochastic processes (ssp), so that **no transients** must be present in the signals
- The input has to **be persistently exciting** of a sufficient order
- Only **input-output models** can be identified

Furthermore, they may optimize a **non-convex** cost function

Adaptive Learning, Estimation and Supervision of dynamical systems

Consider a **closed-loop** SISO control system:

- Usual operation of many plants, in particular if $G_0(z)$ is **open-loop unstable**



- Sometimes the controller is **intrinsically present** (e.g. biomedical, economic systems)
- The controller can have the effect of **«linearizing»** a nonlinear plant

Main (fundamental) difference with the open-loop setting: $u(t)$ and $v(t)$ are **correlated**

Adaptive Learning, Estimation and Supervision of dynamical systems

In most situations the goal of **closed-loop identification** is to estimate $G_0(z)$ and possibly $H_0(z)$. Sometimes, also the estimate of $R(z)$ is of interest. Questions that arise are:

- Are the estimates of $G_0(z)$ and $H_0(z)$ **consistent** when $\mathcal{S} \in \mathcal{M}(\boldsymbol{\theta})$?
- Are the estimates of $G_0(z)$ and $H_0(z)$ **consistent** when only $G_0 \in \mathcal{G}(\boldsymbol{\theta})$?
- Is it possible to **tune the bias** of the estimate?
- What is the **variance** of the estimates?

CLOSED-LOOP IDENTIFICATION METHODS

Adaptive Learning, Estimation and Supervision of dynamical systems

Sometimes, we:

- may have at disposal only signals **with transients**, **not stationary** and **low-order exciting** inputs
- may need a **state-space models** instead of an input-output one (e.g. for Kalman filtering advanced control as **LQR** or **MPC**, model-based **fault diagnosis** approach)

Controllo avanzato multivariabile - 6 CFU

SUBSPACE IDENTIFICATION METHODS

Adaptive Learning, Estimation and Supervision of dynamical systems

Subspace identification approaches are not based on optimization, and they are often used as an **initialization step** for PEM approaches

However, subspace methods allow a simpler identification scheme for **MIMO systems**, as the **number of parameters** to be estimated is often **lower** with respect to PEM approaches

MIMO SYSTEM IDENTIFICATION

Example: estimate a large MIMO system

Suppose that $m_u = 40$ inputs, $p = 28$ outputs, $n = 22$ states

Let an **Input-Output model** assume that each I/O relation is modeled by a **100° order FIR** model. Then, the **total number of parameters** to be estimated is

$$\text{number of params} = m_u \cdot p \cdot 100 = 40 \cdot 28 \cdot 100 = \boxed{112000}$$

State-space model

$$\begin{cases} \underset{n\times 1}{\boldsymbol{x}(t+1)} = \underset{n\times n}{A} \cdot \boldsymbol{x}(t) + \underset{n\times m_u}{B} \cdot \underset{m_u\times 1}{\boldsymbol{u}(t)} \\ \underset{p\times 1}{\boldsymbol{y}(t)} = \underset{p\times n}{C} \cdot \boldsymbol{x}(t) + \underset{p\times m_u}{D} \cdot \boldsymbol{u}(t) \end{cases}$$

$$\begin{aligned} \text{number of params} &= n \cdot n + n \cdot m_u + n \cdot p + p \cdot m_u \\ &= 22 \cdot 22 + 22 \cdot 40 + 22 \cdot 28 + 28 \cdot 40 \\ &= \boxed{3100} \end{aligned}$$

Adaptive Learning, Estimation and Supervision of dynamical systems

C) SUPERVISION OF DYNAMICAL SYSTEMS

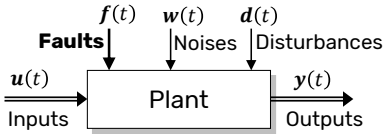
The **supervision (**or **diagnosis)** of dynamical systems aim to **discover anomalies** in the plant, its actuators and sensors, using **measurements** and a **model** of the plant

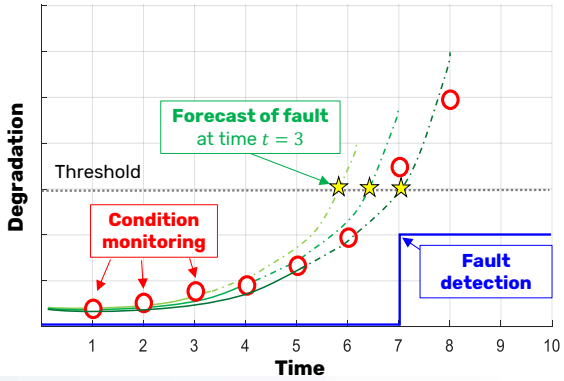
- How to **detect** if a process component is «healthy» or «faulty»?
- How to **isolate** which component is faulty?
- How to be **robust** despite of *noises*, *disturbances* or *model inaccuracies*?

FAULT DIAGNOSIS METHODS

Adaptive Learning, Estimation and Supervision of dynamical systems

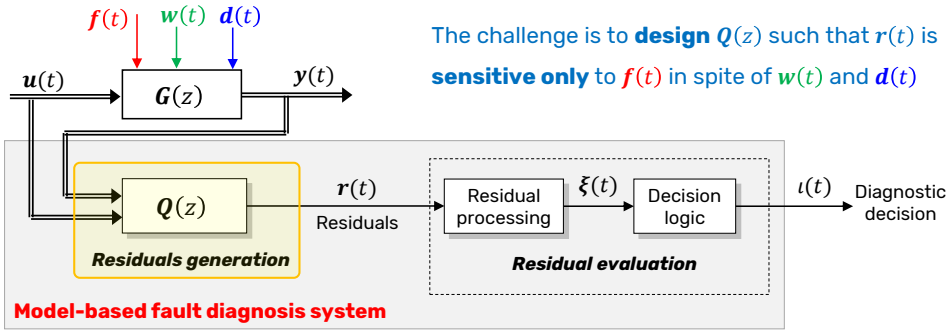
1. Fault **detection** and **isolation**
2. Condition **monitoring**
3. **Prognosis**





Adaptive Learning, Estimation and Supervision of dynamical systems

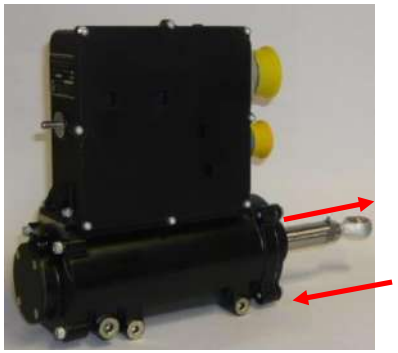
We will mainly focus on the **model-based approach** to fault diagnosis

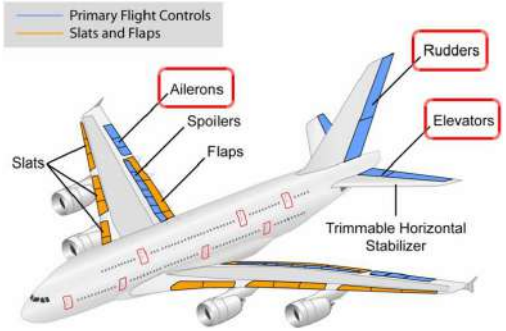


The challenge is to **design** $Q(z)$ such that $r(t)$ is **sensitive only** to $f(t)$ in spite of $w(t)$ and $d(t)$

Example: model-based EMA jamming detection

Consider an EMA actuating **primary flight surfaces:**

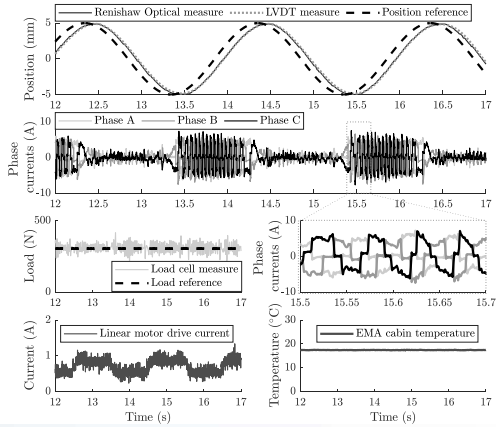




Example: model-based EMA jamming detection

Test bench measurements

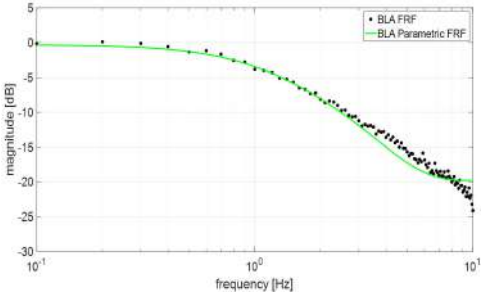
1. EMA Phase currents
2. EMA LVDT position
3. Linear motor position
4. EMA Reference position
5. Load cell



Example: model-based EMA jamming detection

Identification method:

1. **Multisine** excitation
2. **Nonparametric estimate** of the closed-loop transfer function
3. **Parametric estimate** fitting FRF data



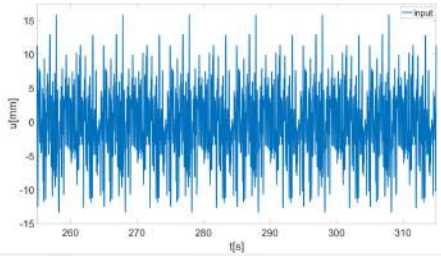
$$\frac{X(z)}{\bar{X}(z)} = G(z) = \frac{0.06892 - 0.1732z^{-1} + 0.1266z^{-2}}{1 - 1.624z^{-1} + 0.6467z^{-2}}$$

- $\bar{x}(t)$: position reference
- $x(t)$: position measure LVDT

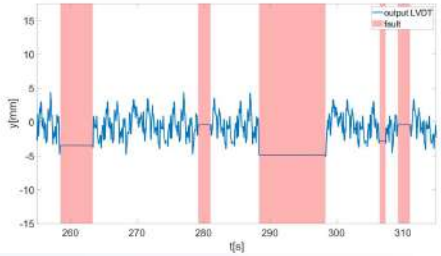
Example: model-based EMA jamming detection

When the degradation took over, the **EMA undergone small-jams** during the operation. The output position signal, as measured by the LVDT sensor, shows **constant value.** We modeled the fault as an **actuator fault**.

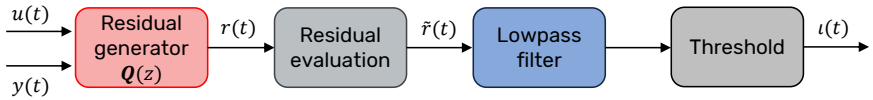
Multisine position reference

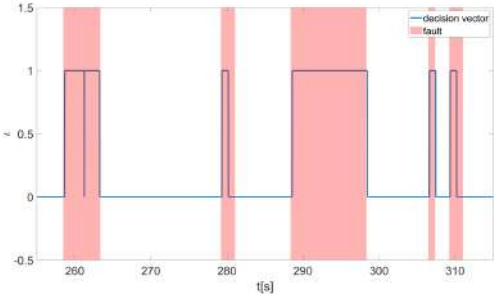


Output position (LVDT)



Example: model-based EMA jamming detection





UNIVERSITÀ DEGLI STUDI DI BERGAMO | Dipartimento di Ingegneria Gestionale, dell'Informazione e della Produzione

UNIVERSITÀ DEGLI STUDI DI BERGAMO | Dipartimento di Ingegneria Gestionale, dell'Informazione e della Produzione

ADAPTIVE LEARNING, ESTIMATION AND SUPERVISION OF DYNAMICAL SYSTEMS (ALES)

Lecture 2: Recursive and adaptive Least Squares (RLS)

Master Degree in COMPUTER ENGINEERING

Data Science and Data Engineering Curriculum

SPEAKER
Prof. Mirko Mazzoleni

PLACE
University of Bergamo

Syllabus

1. **Recursive and adaptive identification**
   - 1.1 Recursive ARX estimation (RLS)
   - 1.2 Least Mean Squares (LMS)
   - 1.3 Instrumental Variables (IV)
2. **Closed-loop identification**
3. **Subspace and MIMO identification**
   - 3.1 Singular Value Decomposition
   - 3.2 Impulse data: Ho-Kalman, Kung algorithms
   - 3.3 Generic I/O data: the MOESP algorithm
4. **Supervision of dynamical systems**
   - 4.1 Introduction to fault diagnosis
   - 4.2 Model-based fault diagnosis
   - 4.3 Parity space approaches
   - 4.4 Observer-based approaches
   - 4.5 Signal-based fault diagnosis
   - 4.6 Knowledge-based fault diagnosis

CASE STUDIES

- Virtual Reference Feedback Tuning
- Nuclear particles classification
- Leak detection in an industrial valve
- Bearing fault identification

Outline

1. Introduction to recursive and adaptive system identification
2. ARX models: Recursive Least Squares (RLS)
3. ARX models: adaptive identification and forgetting factor
4. Connections with Kalman Filtering
5. Least Mean Squares (LMS) algorithm

Outline

1. **Introduction to recursive and adaptive system identification**
2. ARX models: Recursive Least Squares (RLS)
3. ARX models: adaptive identification and forgetting factor
4. Connections with Kalman Filtering
5. Least Mean Squares (LMS) algorithm

Recursive and adaptive identification methods

Recursive and adaptive methods are important in different fields of engineering:

- **Adaptive control:** adapt the **controller** to the current state of the system, that varies according to external factors in an unpredictable way *(airplanes, missiles, automobiles, paper-making machines have dynamic properties that depend on speed, load)*
- **Adaptive prediction:** perform a **prediction** when the system is affected by varying unknown components *(prediction of electricity power demand which depend on whether and other external factors, predictive coding for digital transmission)*
- **Adaptive filtering:** perform a **filtering** operation on signals *(channel equalization, adaptive notch filters)*

Recursive and adaptive identification methods

- **Fault diagnosis:** continuously **monitor** the system measurements to detect changes in its parameters, that might be symptom of faults in the system

In all these cases, a **model** of the system is needed. In particular, the model must be continuously updated in an **on-line** manner, each time a new datum becomes available

In addition to on-line decisions, there are additional benefits of recursive identification:

- **Data compression:** with the on-line processing of data, old data are not required to be stored and can be **discarded**. Given an indication of model accuracy, it is possible to decide when to **stop** data acquisition

Recursive and adaptive identification methods

- **Application to off-line identification:** recursive methods can be efficiently applied to process **batch data** for improving the estimates, e.g. in **smoothing** procedures

The disadvantages of using an on-line approach are:

- The **model family** must be chosen (and then it remains **fixed**) at the beginning of the algorithm
- The **accuracy** of on-line approaches (apart from some cases) is **lower** than that of off-line ones

Recursive and adaptive identification methods

In recursive identification methods the **parameter estimates are adjusted recursively** over time, with an update rule of the following type:

$$\underset{d\times 1}{\hat{\boldsymbol{\theta}}(t)} = \hat{\boldsymbol{\theta}}(t-1) + \text{correction term}$$

To illustrate the derivation of a recursive algorithm, consider the problem of **estimating a constant** from a **noisy measurement**

Example: recursive estimation of a constant

To illustrate the derivation of a recursive algorithm, consider the problem of **estimating a constant from a noisy measurement**

$$\underset{1\times 1}{y(t)} = \underset{1\times 1}{b} + \eta(t)\,, \qquad \eta(t) \sim \mathrm{WN}(0,\lambda^2)$$

The Least Square estimate of $\theta = b$ is given by the **arithmetic mean** over N data

$$\hat{\theta} = \frac{1}{N}\sum_{t=1}^{N} y(t) \quad \Rightarrow$$

This expression can be easily translated into a **recursive form** starting from the LS estimate at a generic time t

$$\hat{\theta}(t) = \frac{1}{t}\sum_{i=1}^{t} y(i) = \frac{1}{t}\left(\sum_{i=1}^{t-1} y(i) + y(t)\right) = \frac{1}{t}\Big((t-1)\cdot\hat{\theta}(t-1) + y(t)\Big) = \hat{\theta}(t-1) + \frac{1}{t}\Big(y(t) - \hat{\theta}(t-1)\Big)$$

Example: recursive estimation of a constant

To compute the estimate $\hat{\theta}(t)$ at time t, all we need is:

- the **estimate** at the **previous time** instant, $\hat{\theta}(t-1)$
- the **current datum** $y(t)$

Remarks:

- the **correction term** with respect to the previous estimate is proportional to the difference between the actual observed value $y(t)$ and the predicted value $\hat{\theta}(t-1)$ **(prediction error)**
- the **prediction error** is weighted by $1/t$
  - ✓ the corrections to the estimate get smaller as time goes by
  - ✓ as the number of data increases, the estimate $\hat{\theta}(t-1)$ becomes more reliable

Outline

Recursive Least Squares (RLS), form 1

Consider a generic SISO $\mathrm{ARX}(n_a, n_b, k = 1)$ model:

$$\mathcal{S}: \; y(t) = a_1 y(t-1) + \cdots + a_{n_a} y(t-n_a) + b_0 u(t-1) + \cdots + b_{n_b} u(t-n_b-1) + e(t)$$

$$= \underset{1\times d}{\boldsymbol{\varphi}^\top(t)} \underset{d\times 1}{\boldsymbol{\theta}} + e(t)$$

- $\boldsymbol{\varphi}(t) = [y(t-1) \quad \cdots \quad y(t-n_a) \quad u(t-1) \quad \cdots \quad u(t-n_b-1)]^\top$
- $\boldsymbol{\theta} = [a_1 \quad \cdots \quad a_{n_a} \quad b_1 \quad \cdots \quad b_{n_b}]^\top$
- $e(t) \sim \mathrm{WN}(0, \lambda^2)$ is an unknown white-noise signal

We already have a solution to the **batch identification problem**, i.e. the optimal estimation of the parameter vector $\boldsymbol{\theta}$, when a batch of N data is available:

- **Input data:** $\{u(1), u(2), \ldots, u(N)\}$
- **Output data:** $\{y(1), y(2), \ldots, y(N)\}$

Recursive Least Squares (RLS), form 1

According to the **predictive approach** (PEM method), the optimal parameters estimate minimizes a quadratic cost function of the prediction error

Given a $\boldsymbol{\theta}$, the **best predictor** of $y(t)$ we can provide at time $t-1$ is $\hat{y}(t|t-1;\boldsymbol{\theta}) = \boldsymbol{\varphi}^\top(t)\boldsymbol{\theta}$

Notice here that $\boldsymbol{\varphi}(t)$ **depends on data up to time** $t-1$ **only**

The corresponding **prediction error** is $\varepsilon(t;\boldsymbol{\theta}) = y(t) - \hat{y}(t|t-1;\boldsymbol{\theta}) = y(t) - \boldsymbol{\varphi}^\top(t)\boldsymbol{\theta}$

Accordingly, the **optimal parameters estimate** corresponds to $\hat{\boldsymbol{\theta}}_N = \mathrm{argmin}_{\boldsymbol{\theta}} J_N(\boldsymbol{\theta})$

where $J_N(\boldsymbol{\theta}) = \frac{1}{N}\sum_{t=1}^{N} \varepsilon(t;\boldsymbol{\theta})^2$ is the **sampled variance of the prediction error**

Recursive Least Squares (RLS), form 1

Since $J_N(\boldsymbol{\theta})$ is a quadratic performance index, one can obtain the solution by imposing:

$$-\frac{2}{N}\sum_{t=1}^{N} \underset{d\times 1}{\boldsymbol{\varphi}(t)} \left(y(t) - \underset{1\times d}{\boldsymbol{\varphi}^\top(t)} \underset{d\times 1}{\boldsymbol{\theta}}\right) = \underset{d\times 1}{\mathbf{0}}$$

Transposing and separating terms, one obtains the **normal equations:**

$$\hat{\boldsymbol{\theta}} = \left(\sum_{t=1}^{N} \underset{d\times 1}{\boldsymbol{\varphi}(t)} \underset{1\times d}{\boldsymbol{\varphi}^\top(t)}\right)^{-1} \cdot \sum_{t=1}^{N} \underset{d\times 1}{\boldsymbol{\varphi}(t)} \underset{1\times 1}{y(t)}$$

How can we **update** (and improve) the **estimate based on new data arriving**, without reapplying the batch formula from scratch every time?

Recursive Least Squares (RLS), form 1

Consider the batch LS formula applied to the data up to time t

$$\hat{\boldsymbol{\theta}}(t) = \underset{d\times d}{S(t)^{-1}} \cdot \sum_{i=1}^{t} \underset{d\times 1}{\boldsymbol{\varphi}(i)} \underset{1\times 1}{y(i)} \quad \text{where} \quad S(t) = \sum_{i=1}^{t} \underset{d\times 1}{\boldsymbol{\varphi}(i)} \underset{1\times d}{\boldsymbol{\varphi}^\top(i)}$$

Both $S(t)$ and $\sum_{i=1}^{t} \boldsymbol{\varphi}(i)y(i)$ **can be rewritten in recursive form:**

- $S(t) = \sum_{i=1}^{t-1} \boldsymbol{\varphi}(i)\boldsymbol{\varphi}^\top(i) + \boldsymbol{\varphi}(t)\boldsymbol{\varphi}^\top(t) = S(t-1) + \boldsymbol{\varphi}(t)\boldsymbol{\varphi}^\top(t)$
- $\sum_{i=1}^{t} \boldsymbol{\varphi}(i)y(i) = \sum_{i=1}^{t-1} \boldsymbol{\varphi}(i)y(i) + \boldsymbol{\varphi}(t)y(t)$

The LS estimate at time $t-1$ is equal to: $\hat{\boldsymbol{\theta}}(t-1) = S(t-1)^{-1} \cdot \sum_{i=1}^{t-1} \boldsymbol{\varphi}(i)y(i)$

Recursive Least Squares (RLS), form 1

Putting all these expressions together we obtain:

$$\underset{d\times 1}{\hat{\boldsymbol{\theta}}(t)} = \underset{d\times d}{S(t)^{-1}} \cdot \sum_{i=1}^{t} \underset{d\times 1}{\boldsymbol{\varphi}(i)} \underset{1\times 1}{y(i)} = \underset{d\times d}{S(t)^{-1}} \cdot \left(\sum_{i=1}^{t-1} \boldsymbol{\varphi}(i)y(i) + \underset{d\times 1}{\boldsymbol{\varphi}(t)} \underset{1\times 1}{y(t)} \right)$$

$$= S(t)^{-1} \cdot \left(S(t-1)\hat{\boldsymbol{\theta}}(t-1) + \boldsymbol{\varphi}(t)y(t) \right) = S(t)^{-1} \cdot \left((S(t) - \boldsymbol{\varphi}(t)\boldsymbol{\varphi}(t)^{\mathsf{T}})\hat{\boldsymbol{\theta}}(t-1) + \boldsymbol{\varphi}(t)y(t) \right)$$

$$= \hat{\boldsymbol{\theta}}(t-1) + \underbrace{S(t)^{-1}\boldsymbol{\varphi}(t)}_{\text{Gain vector } K(t),\ d\times 1} \underbrace{\left(y(t) - \boldsymbol{\varphi}^{\mathsf{T}}(t)\hat{\boldsymbol{\theta}}(t-1) \right)}_{\text{Prediction error } \varepsilon(t),\ 1\times 1}$$

$$\Rightarrow \quad \underset{d\times 1}{\hat{\boldsymbol{\theta}}(t)} = \underset{d\times 1}{\hat{\boldsymbol{\theta}}(t-1)} + \underset{d\times 1}{K(t)} \cdot \underset{1\times 1}{\varepsilon(t)}$$

Recursive Least Squares (RLS), form 1

RLS form 1 summary

- Basic recursion: $\hat{\boldsymbol{\theta}}(t) = \hat{\boldsymbol{\theta}}(t-1) + K(t) \cdot \varepsilon(t)$
- Definitions: $K(t) = S(t)^{-1}\boldsymbol{\varphi}(t)$

  $\varepsilon(t) = y(t) - \boldsymbol{\varphi}^{\mathsf{T}}(i)\hat{\boldsymbol{\theta}}(t-1)$
- Auxiliary recursion: $S(t) = S(t-1) + \boldsymbol{\varphi}(t)\boldsymbol{\varphi}^{\mathsf{T}}(i)$

Recursive Least Squares (RLS), form 1

Remarks:

- $\varepsilon(t)$ is the **difference** between the **measured** output $y(t)$ and the one-step-ahead **prediction** $\hat{y}\left(t|t-1;\hat{\boldsymbol{\theta}}(t-1)\right)$ made at time $t-1$, based on the model associated to the estimated parameters $\hat{\boldsymbol{\theta}}(t-1)$ ⇨ *prediction error*
- If $\varepsilon(t)$ is **small**, the current **model is «good»** and must not to be changed much
- If the prediction error $\varepsilon(t)$ is **zero** at time t, the update term $K(t)\varepsilon(t)$ will be zero and the parameter **estimate** will **not be modified**
- On the other hand, if $\varepsilon(t) \neq 0$ the estimate is modified by a quantity that is proportional to $\varepsilon(t)$ through a factor $K(t)$ (**«gain»** of the algorithm)

Example: recursive estimation of a constant

The considered model is an $\mathrm{ARX}(n_a = 0, n_b = 0, k = 1)$

$$y(t) = bu(t-1) + \eta(t), \quad \eta(t) \sim \mathrm{WN}(0,1), \ b = 3$$

with $u(t-1) = 1, \forall t \geq 0$. Accordingly, it can be reformulated as a **linear regression:**

$$y(t) = \boldsymbol{\varphi}^{\mathsf{T}}(t)\boldsymbol{\theta} + \eta(t) \quad \text{with } \boldsymbol{\varphi}(t) = u(t-1) = 1 \text{ and } \theta = b.$$

Now, since $S(t) = S(t-1) + \boldsymbol{\varphi}(t)\boldsymbol{\varphi}^{\mathsf{T}}(t) = S(t-1) + 1$ and $S(1) = 1$, we obtain $S(t) = t$

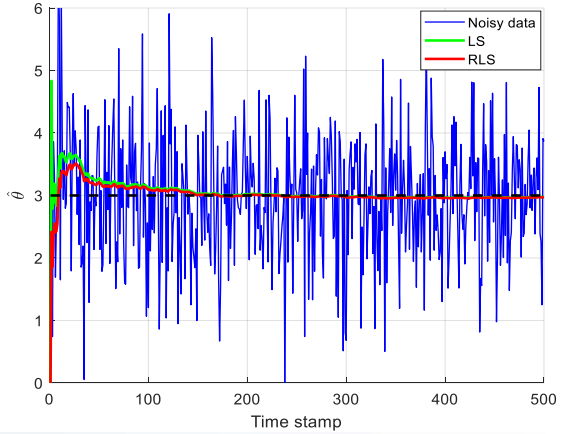
Consequently, $K(t) = S(t)^{-1}\boldsymbol{\varphi}(t) = \frac{1}{t}$. Since $\varepsilon(t) = y(t) - \boldsymbol{\varphi}^{\mathsf{T}}(t)\hat{\boldsymbol{\theta}}(t-1) = y(t) - \hat{\boldsymbol{\theta}}(t-1)$, we have that:

$$\hat{\boldsymbol{\theta}}(t) = \hat{\boldsymbol{\theta}}(t-1) + K \cdot \varepsilon(t) = \hat{\boldsymbol{\theta}}(t-1) + \frac{1}{t}\left(y(t) - \hat{\boldsymbol{\theta}}(t-1)\right)$$

as previously described.

Example: recursive estimation of a constant

The RLS algorithm converges to the **same result** of the LS method



Recursive Least Squares (RLS), form 2

In stationary conditions, $S(t)$ **diverges to infinity** for $t \to +\infty$ (much before that, it will saturate the numerical precision of the digital computing unit)

The previous problem can be solved by resorting to the matrix $R(t) = \frac{S(t)}{t}$, instead of the plain $S(t)$. Indeed, $R(t)$ is the (sampled) **auto-correlation matrix** associated to vector $\boldsymbol{\varphi}(t)$, which **converges in stationarity conditions**

$$R(t) = \frac{1}{t}S(t) = \frac{1}{t}\left(S(t-1) + \boldsymbol{\varphi}(t)\boldsymbol{\varphi}^{\top}(t)\right) = \frac{t-1}{t}\cdot\frac{1}{t-1}S(t-1) + \frac{1}{t}\boldsymbol{\varphi}(t)\boldsymbol{\varphi}^{\top}(t)$$

$$= \frac{t-1}{t}\cdot R(t-1) + \frac{1}{t}\boldsymbol{\varphi}(t)\boldsymbol{\varphi}^{\top}(t) = R(t-1) + \frac{1}{t}\left(\boldsymbol{\varphi}(t)\boldsymbol{\varphi}^{\top}(t) - R(t-1)\right)$$

Recursive Least Squares (RLS), form 2

RLS form 2 summary

- Basic recursion: $\hat{\boldsymbol{\theta}}(t) = \hat{\boldsymbol{\theta}}(t-1) + K(t)\cdot\varepsilon(t)$
- Definitions: $K(t) = \frac{1}{t}R(t)^{-1}\boldsymbol{\varphi}(t)$

  $\varepsilon(t) = y(t) - \boldsymbol{\varphi}^{\top}(t)\hat{\boldsymbol{\theta}}(t-1)$
- Auxiliary recursion: $R(t) = R(t-1) + \frac{1}{t}\left(\boldsymbol{\varphi}(t)\boldsymbol{\varphi}^{\top}(t) - R(t-1)\right)$

- Form 2 is an improvement over form 1 in that $R(t)$ does not diverge with t, as does $S(t)$
- Both forms have the **drawback** that at each step an $d \times d$ matrix inversion is required

Can the matrix inversion be avoided?

Recursive Least Squares (RLS), form 3

To solve the mentioned problem, let's resort to the **matrix inversion lemma** (or Householder's lemma), which states that

$$(F + GHK)^{-1} = F^{-1} - F^{-1}G(H^{-1} + KF^{-1}G)^{-1}KF^{-1}$$

provided that all the inverse matrices appearing in the previous expression exist.

In our case, we have:

$$\underset{d\times d}{S(t)^{-1}} = (\underbrace{S(t-1)}_{F\ d\times d} + \underbrace{\boldsymbol{\varphi}(t)}_{G\ d\times 1}\cdot\underbrace{1}_{H\ 1\times 1}\cdot\underbrace{\boldsymbol{\varphi}(t)^{\top}}_{K\ 1\times d})^{-1}$$

So that we can write

$$\underset{d\times d}{S(t)^{-1}} = \underset{d\times d}{S(t-1)^{-1}} - \underset{d\times 1}{S(t-1)^{-1}\boldsymbol{\varphi}(t)}\left(1 + \underset{1\times d}{\boldsymbol{\varphi}^{\top}(t)}\underset{d\times d}{S(t-1)^{-1}}\underset{d\times 1}{\boldsymbol{\varphi}(t)}\right)^{-1}\underset{1\times d}{\boldsymbol{\varphi}^{\top}(t)S(t-1)^{-1}}$$

Recursive Least Squares (RLS), form 3

This implies that, once $S(t-1)^{-1}$ is given, calculating $S(t)^{-1}$ is only a matter of **inverting a scalar** (and performing a few matrix multiplications)

Defining $P(t) = S(t)^{-1}$ we obtain the auxiliary recursion

$$P(t) = P(t-1) - \frac{P(t-1)\boldsymbol{\varphi}(t)\boldsymbol{\varphi}^{\mathsf{T}}(t)P(t-1)}{1+\boldsymbol{\varphi}^{\mathsf{T}}(t)P(t-1)\boldsymbol{\varphi}(t)}$$

$$K(t) = P(t)\boldsymbol{\varphi}(t)$$

Recursive Least Squares (RLS), form 3

The **computation of the gain** $K(t) = P(t)\boldsymbol{\varphi}(t)$ can be simplified as follows:

$$\underset{d\times 1}{K(t)} = \underset{d\times d}{P(t)}\underset{d\times 1}{\boldsymbol{\varphi}(t)} = \left[P(t-1) - \frac{P(t-1)\boldsymbol{\varphi}(t)\boldsymbol{\varphi}^{\mathsf{T}}(t)P(t-1)}{1+\boldsymbol{\varphi}^{\mathsf{T}}(t)P(t-1)\boldsymbol{\varphi}(t)}\right]\boldsymbol{\varphi}(t)$$

$$= P(t-1)\boldsymbol{\varphi}(t) - \frac{P(t-1)\boldsymbol{\varphi}(t)\boldsymbol{\varphi}^{\mathsf{T}}(t)P(t-1)\boldsymbol{\varphi}(t)}{1+\boldsymbol{\varphi}^{\mathsf{T}}(t)P(t-1)\boldsymbol{\varphi}(t)}$$

$$= \frac{P(t-1)\boldsymbol{\varphi}(t)\left(1+\boldsymbol{\varphi}^{\mathsf{T}}(t)P(t-1)\boldsymbol{\varphi}(t)\right)}{1+\boldsymbol{\varphi}^{\mathsf{T}}(t)P(t-1)\boldsymbol{\varphi}(t)} - \frac{P(t-1)\boldsymbol{\varphi}(t)\boldsymbol{\varphi}^{\mathsf{T}}(t)P(t-1)\boldsymbol{\varphi}(t)}{1+\boldsymbol{\varphi}^{\mathsf{T}}(t)P(t-1)\boldsymbol{\varphi}(t)}$$

$$= \frac{P(t-1)\boldsymbol{\varphi}(t)}{1+\boldsymbol{\varphi}^{\mathsf{T}}(t)P(t-1)\boldsymbol{\varphi}(t)}$$

Recursive Least Squares (RLS), form 3

$$\underset{d\times 1}{K(t)} = \underset{d\times d}{P(t)}\underset{d\times 1}{\boldsymbol{\varphi}(t)} = \frac{P(t-1)\boldsymbol{\varphi}(t)}{1+\boldsymbol{\varphi}^{\mathsf{T}}(t)P(t-1)\boldsymbol{\varphi}(t)}$$

Thus, the gain $K(t)$ can be computed using the quantity

$$\frac{P(t-1)\boldsymbol{\varphi}(t)}{1+\boldsymbol{\varphi}^{\mathsf{T}}(t)P(t-1)\boldsymbol{\varphi}(t)}$$

which is **already available** when updating the matrix $P(t)$

Recursive Least Squares (RLS), form 3

RLS form 3 summary

- Basic recursion: $\underset{d\times 1}{\widehat{\boldsymbol{\theta}}(t)} = \widehat{\boldsymbol{\theta}}(t-1) + K(t)\cdot\varepsilon(t)$
- Definitions: $\underset{d\times 1}{K(t)} = \underset{d\times d}{P(t)}\underset{d\times 1}{\boldsymbol{\varphi}(t)} = \frac{P(t-1)\boldsymbol{\varphi}(t)}{1+\boldsymbol{\varphi}^{\mathsf{T}}(t)P(t-1)\boldsymbol{\varphi}(t)}$

  $\underset{1\times 1}{\varepsilon(t)} = y(t) - \underset{1\times d}{\boldsymbol{\varphi}^{\mathsf{T}}(t)}\underset{d\times 1}{\widehat{\boldsymbol{\theta}}(t-1)}$
- Auxiliary recursion: $\underset{d\times d}{P(t)} = P(t-1) - \beta(t-1)^{-1}\cdot P(t-1)\boldsymbol{\varphi}(t)\boldsymbol{\varphi}^{\mathsf{T}}(t)P(t-1)$

  $\underset{1\times 1}{\beta(t-1)} = 1 + \underset{1\times d}{\boldsymbol{\varphi}^{\mathsf{T}}(t)}\underset{d\times d}{P(t-1)}\underset{d\times 1}{\boldsymbol{\varphi}(t)}$

Properties of the matrices $S(t), R(t), P(t)$

$S(t), R(t)$ and $P(t)$ are symmetric **semi-definite positive** matrices, by construction. By assuming that $S(t)$ is **invertible**, all matrices are also **definite positive** *(the inverse of a definite positive matrix is definite positive as well)*

When inputs and outputs **are stationary**, we expect that, for $t \to +\infty$:

- $S(t) = \sum_{i=1}^{t} \boldsymbol{\varphi}(i)\boldsymbol{\varphi}^{\top}(i)$ diverges
- $R(t) = S(t)/t$ asymptotically converges to $\bar{R}$
- $P(t) = S(t)^{-1}$ tends to 0

In fact, by construction, the difference between two successive values of $P(t)$ is **negative semidefinite:** $P(t) - P(t-1) = -\beta(t-1)^{-1} \cdot P(t-1)\boldsymbol{\varphi}(t)\boldsymbol{\varphi}^{\top}(t)P(t-1) \preccurlyeq 0$

Properties of the matrices $S(t), R(t), P(t)$

IMAD - Lecture 3, Slide 25

Therefore, $P(t)$ **can only decrease or stay constant** (the latter behaviour occurs if $\boldsymbol{\varphi}(t) = \mathbf{0}$) which means that the last datum does not bring any information on $\boldsymbol{\theta}$

Recall from the analysis of the **batch LS algorithm** that $\operatorname{var}\left[\widehat{\boldsymbol{\theta}}(N)\right] = \lambda^2 S(N)^{-1} = \lambda^2 P(N)$

Thus, apart from a constant factor, matrix $P(t)$ represents the **covariance estimation matrix**, which tends asymptotically to 0 as $1/N$. Accordingly, we can interpret $S(t)$ as an **information matrix**. Thus:

- $P(t)$ **decreases** → the uncertainty in the parameters decreases with time
- $S(t)$ **increases** → the available information grows as the number of data increases
- $K(t)$ **decreases** → the prediction errors are less and less considered to adjust the model, because they are more and more attributed to noise

Properties of the matrices $S(t), R(t), P(t)$

Remarks:

- the **batch method** requires the invertibility of $S(N)$ for identifiability; on the other hand, the **recursive method**, that no longer involves matrix inversion, will **not be able to detect unidentifiability** and will converge to a particular solution that depends on the initialization
- since $\lambda^2 P(t)$ characterizes the **uncertainty in the estimated parameters**, it is important to **monitor its evolution**, or at least that of its diagonal terms
- By defining $P^*(t) = \lambda^2 P(t)$ as the covariance matrix of the parameters estimates, the recursive expressions modify in the **stochastic RLS algorithm**

Properties of the matrices $S(t), R(t), P(t)$

Stochastic RLS form 3 summary

- Basic recursion: $\underset{d\times 1}{\widehat{\boldsymbol{\theta}}(t)} = \underset{d\times 1}{\widehat{\boldsymbol{\theta}}(t-1)} + \underset{d\times 1}{K(t)} \cdot \varepsilon(t)$
- Definitions: $\underset{d\times 1}{K(t)} = \frac{\underset{d\times d}{P^*(t)}}{\lambda^2} \underset{d\times 1}{\boldsymbol{\varphi}(t)} = \frac{P^*(t-1)\boldsymbol{\varphi}(t)}{\lambda^2 + \boldsymbol{\varphi}^{\top}(t)P^*(t-1)\boldsymbol{\varphi}(t)}$

  $\underset{1\times 1}{\varepsilon(t)} = y(t) - \boldsymbol{\varphi}^{\top}(t)\widehat{\boldsymbol{\theta}}(t-1)$
- Auxiliary recursion: $\underset{d\times d}{P^*(t)} = P^*(t-1) - \beta(t-1)^{-1} \cdot P^*(t-1)\boldsymbol{\varphi}(t)\boldsymbol{\varphi}^{\top}(t)P^*(t-1)$

  $\underset{1\times 1}{\beta(t-1)} = \lambda^2 + \underset{1\times d}{\boldsymbol{\varphi}^{\top}(t)} \underset{d\times 1}{P^*(t-1)\boldsymbol{\varphi}(t)}$

RLS algorithm initialization

The RLS (in all its forms) is **an exact recursive version** of the batch LS algorithm. This implies that with a suitable initialization the RLS **will provide the same results** (on the same set of data) of its batch counterpart.

An **exact initialization** would be (where t_0 is a time instant where $S(t_0)$ is invertible)

$$\underset{d\times d}{P(t_0)} = \left(\sum_{i=1}^{t_0} \underset{1\times d}{\boldsymbol{\varphi}(i)}\underset{1\times d}{\boldsymbol{\varphi}^{\top}(i)}\right)^{-1} = S(t_0)^{-1} \qquad \underset{d\times 1}{\hat{\boldsymbol{\theta}}(t_0)} = \underset{d\times d}{P(t_0)} \sum_{i=1}^{t_0} \underset{d\times 1}{\boldsymbol{\varphi}(i)}\underset{1\times 1}{y(i)}$$

A **simpler initialization** can be employed, setting $P(0)$ **large** (10^6) and $\hat{\boldsymbol{\theta}}(0)$ **arbitrarily**, the only condition being that $P(0)$ be a **symmetric positive definite matrix** *($P(t)$ must be positive definite at all times)*

Outline

Adaptive identification

As discussed, in the stationary case $\underset{d\times d}{P(t)} = \left(\sum_{i=1}^{t} \underset{d\times 1}{\boldsymbol{\varphi}(i)}\underset{1\times d}{\boldsymbol{\varphi}^{\top}(i)}\right)^{-1}$ tends to 0 for $t \to +\infty$, which implies that the **parameter estimate will stop to change** after enough data has been processed

This is **fine for constant** parameters, but what if they **change in time?** This may occur if:

- the system is **linear but time-varying**
- the system is **time invariant, but nonlinear**

RLS (as we derived it) is **not suitable for adaptive identification** purposes, because, due to the divergence of $S(t)$ *(or equivalently the convergence of $P(t)$ to 0)*, the gain $K(t)$ of the algorithm will be inevitably driven to 0, switching progressively off the algorithm

Adaptive identification

We need to keep the algorithm **reactive to parameter variations** *(which modify the prediction error $\varepsilon(t)$)*, by avoiding $K(t)$ from going to 0. To achieve this, we must **forget older measurements**, that are no more representative of the current situation

The trick is to **discount the importance of long past data**, so that the estimation is mostly influenced by recent data

This can be obtained by minimizing a **modified cost:**

$$J_N^{\mu}(\boldsymbol{\theta}) = \frac{1}{N}\sum_{t=1}^{N} \mu^{N-t} \cdot (y(t) - \boldsymbol{\varphi}^{\top}(t)\boldsymbol{\theta})^2, \qquad 0 < \mu \leq 1$$

- Parameter μ is called the **forgetting factor.** The **smaller** it is, the **quicker** will past data be discounted

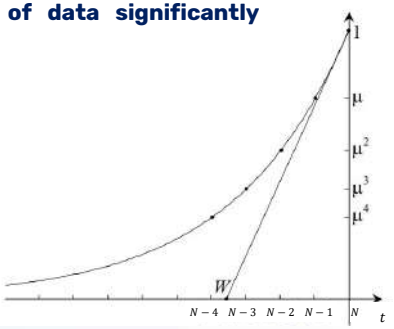
Adaptive identification

The quantity $W = 1/(1-\mu)$ (algorithm *window*) can be interpreted as the time constant associated to the discrete exponential μ^{N-t} ($W^o = -1/\log(\mu)$)

It provides a rough estimate of the **number of data significantly weighted** by the algorithm

- $\mu = 0.99 \quad W = 100$ data
- $\mu = 0.98 \quad W = 50$ data
- $\mu = 0.96 \quad W = 25$ data

Smaller μ ⇨ Data **forgotten quicker**



Adaptive identification

Since $J_N^{\mu}(\boldsymbol{\theta})$ is a quadratic performance index, one can obtain the solution by imposing:

$$-\frac{2}{N}\sum_{t=1}^{N}\mu^{N-t}\boldsymbol{\varphi}(t)(y(t)-\boldsymbol{\varphi}^{\mathsf{T}}(t)\boldsymbol{\theta}) = \mathbf{0}$$

Transposing and separating terms, one obtains the **normal equations:**

$$\widehat{\boldsymbol{\theta}} = \left(\sum_{t=1}^{N}\mu^{N-t}\boldsymbol{\varphi}(t)\boldsymbol{\varphi}^{\mathsf{T}}(t)\right)^{-1}\cdot\sum_{t=1}^{N}\mu^{N-t}\boldsymbol{\varphi}(t)y(t)$$

The recursive formulation can be found as in the standard RLS case

Adaptive identification

$$\widehat{\boldsymbol{\theta}} = \left(\sum_{t=1}^{N}\mu^{N-t}\boldsymbol{\varphi}(t)\boldsymbol{\varphi}^{\mathsf{T}}(t)\right)^{-1}\cdot\sum_{t=1}^{N}\mu^{N-t}\boldsymbol{\varphi}(t)y(t)$$

Let's define the following quantities

$$S(t) = \sum_{i=1}^{t}\mu^{t-i}\boldsymbol{\varphi}(i)\boldsymbol{\varphi}^{\mathsf{T}}(i) \quad = \sum_{i=1}^{t-1}\mu^{t-i}\boldsymbol{\varphi}(i)\boldsymbol{\varphi}^{\mathsf{T}}(i) + \boldsymbol{\varphi}(t)\boldsymbol{\varphi}^{\mathsf{T}}(t)$$

$$S(t-1) = \sum_{i=1}^{t-1}\mu^{t-1-i}\boldsymbol{\varphi}(i)\boldsymbol{\varphi}^{\mathsf{T}}(i) \quad \Rightarrow \quad \boxed{S(t) = \mu S(t-1) + \boldsymbol{\varphi}(t)\boldsymbol{\varphi}^{\mathsf{T}}(t)}$$

Adaptive identification

By grouping μ, it follows that $S(t)$ can be expressed as

$$S(t) = \mu S(t-1) + \boldsymbol{\varphi}(t)\boldsymbol{\varphi}^{\mathsf{T}}(t) \; = \mu\left(S(t-1) + \boldsymbol{\varphi}(t)\frac{1}{\mu}\boldsymbol{\varphi}^{\mathsf{T}}(t)\right)$$

$$S(t)^{-1} = \frac{1}{\mu}\left(S(t-1) + \boldsymbol{\varphi}(t)\frac{1}{\mu}\boldsymbol{\varphi}^{\mathsf{T}}(t)\right)^{-1}$$

$$(F+GHK)^{-1} = F^{-1} - F^{-1}G(H^{-1}+KF^{-1}G)^{-1}KF^{-1}$$

$$= \frac{1}{\mu}\left(S(t-1)^{-1} - S(t-1)^{-1}\boldsymbol{\varphi}(t)\left(\mu + \boldsymbol{\varphi}^{\mathsf{T}}(t)S(t-1)^{-1}\boldsymbol{\varphi}(t)\right)^{-1}\boldsymbol{\varphi}^{\mathsf{T}}(t)S(t-1)^{-1}\right)$$

Thus

$$\boxed{P(t) = \frac{1}{\mu}\left(P(t-1) - \frac{P(t-1)\boldsymbol{\varphi}(t)\boldsymbol{\varphi}^{\mathsf{T}}(t)P(t-1)}{\mu + \boldsymbol{\varphi}^{\mathsf{T}}(t)P(t-1)\boldsymbol{\varphi}(t)}\right)}$$

Adaptive identification

Adaptive RLS form 3 summary

- Basic recursion: $\underset{d\times 1}{\widehat{\boldsymbol{\theta}}(t)} = \widehat{\boldsymbol{\theta}}(t-1) + K(t)\cdot\varepsilon(t)$
- Definitions: $\underset{d\times 1}{K(t)} = \underset{d\times d}{P(t)}\underset{d\times 1}{\boldsymbol{\varphi}(t)} = \dfrac{P(t-1)\boldsymbol{\varphi}(t)}{\mu + \boldsymbol{\varphi}^{\top}(t)P(t-1)\boldsymbol{\varphi}(t)}$

  $\underset{1\times 1}{\varepsilon(t)} = y(t) - \boldsymbol{\varphi}^{\top}(t)\widehat{\boldsymbol{\theta}}(t-1)$
- Auxiliary recursion: $\underset{d\times d}{P(t)} = \dfrac{1}{\mu}[P(t-1) - \underset{1\times 1}{\beta(t-1)^{-1}}\cdot P(t-1)\boldsymbol{\varphi}(t)\boldsymbol{\varphi}^{\top}(t)P(t-1)]$

  $\underset{1\times 1}{\beta(t-1)} = \mu + \underset{1\times d}{\boldsymbol{\varphi}^{\top}(t)}P(t-1)\underset{d\times 1}{\boldsymbol{\varphi}(t)}$

Adaptive identification

Remark: the effect of the forgetting factor μ is to **keep large** $P(t)$, and thus the gain $K(t)$, so that parameters are **allowed to continuously change**

Forgetting (or leakeage) factor

The recursive equation for $S(t)$ can be interpreted as a **state equation of a dynamical system**, where $S(t)$ has the role of the state variable and $\boldsymbol{\varphi}(t)\boldsymbol{\varphi}(t)^{\top}$ is the input:

- the original **recursion is a discrete integrator** (not asymptotically stable), so that $S(t)$ can diverge for stationary inputs
- the modified recursion, with $0 < \mu < 1$, makes this dynamical system **asymptotically stable**, so that $S(t)$ does not diverge and $K(t)$ does not tend to 0

There are some unwanted consequences as well:

- we pay for the increased reactivity of the algorithm with a **distortion in the optimal estimates** in the stationary case (due to the modification in the cost function induced by the leakage factor)

Example: forgetting factor

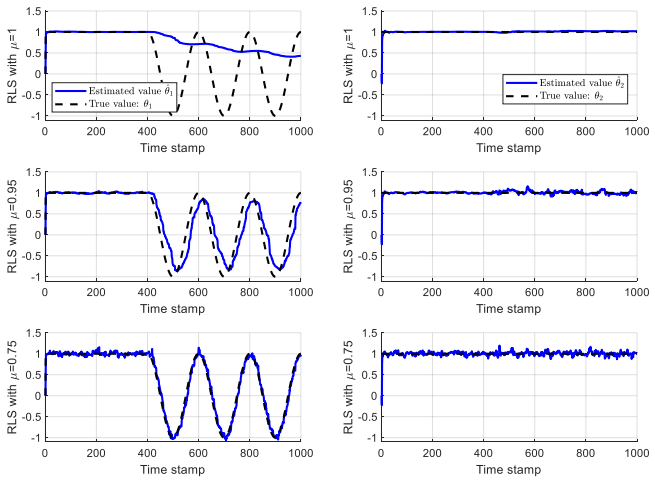
Assume that the system generating the data is given as follows:

$S: y(t) = \theta_1(t)u_1(t) + \theta_2 u_2(t) + e(t)$

- $\theta_1(t) = \begin{cases} 1 & 1 \le t < 400 \\ \sin(0.01\pi t + \pi/2), & 400 \le t \le 1000 \end{cases}$
- $\theta_2 = 1$
- $u_1(t), u_2(t) \sim \mathrm{WN}(0,100)$, with $u_1 \perp u_2$
- $e(t) \sim \mathrm{WN}(0,1)$

The following pictures show how the two parameters are estimated using:

- Plain RLS
- RLS with forgetting factor $\mu = 0.95$
- RLS with forgetting factor $\mu = 0.75$



RLS

RLS $\mu = 0.95$

RLS $\mu = 0.75$

Example: forgetting factor

Remarks:

- the **reactivity** of the algorithm **improves as** μ **decreases**
- the **parameter estimates are more erratic** as μ **decreases**, because the estimation is carried out on a smaller amount of data $W = \frac{1}{1-\mu}$ thus increasing the uncertainty of the estimates

How to choose μ**?**

- $\mu \to 1$: large W, **low** tracking **speed**, **high precision** (small estimation variance)
- $\mu \to 0$: short W, **high** tracking **speed**, **low precision** (large estimation variance)

Time-varying forgetting factor

Advanced material

In practice, it has been found advisable to replace the constant μ by a **variable forgetting factor** $\mu(t)$ which is initially smaller than μ but approaches it asymptotically

This aids initial convergence by ensuring that the **effects of initial conditions** on the recursions are **rapidly removed**

- This strategy can be employed also when the **parameters are constant**

$$\underset{1\times 1}{\mu(t)} = \rho \cdot \underset{1\times 1}{\mu(t-1)} + (1-\rho)\bar{\mu}, \qquad \mu(0) = \mu_0$$

Usually, $\bar{\mu} \approx 1$ in order to fully exploit the information available with the data, with $0 < \mu < 1$, so that the dynamics of $\mu(t)$ are asymptotically stable and converge to the only equilibrium $\bar{\mu} = 1$, with a speed that can be precisely set through ρ

Time-varying forgetting factor

Advanced material

Another strategy for a time-variant forgetting factor, that does not require to set ρ and $\mu(0)$, is to let:

$$\underset{1\times 1}{\mu(t)} = 1 - \rho(t) \qquad \underset{1\times 1}{\rho(t)} = \frac{1-\bar{\mu}}{1-\bar{\mu}^{t+1}}$$

A time-varying forgetting factor is useful also to estimate the **noise variance** λ^2 as:

$$\underset{1\times 1}{\hat{\lambda}^2(t)} = \hat{\lambda}^2(t-1) + p(t)\left[\varepsilon^2(t) - \hat{\lambda}^2(t-1)\right]$$

$$\underset{1\times 1}{p(t)} = \frac{1}{\mu(t)}\left[p(t-1) - \frac{p^2(t-1)}{\mu(t)+p(t-1)}\right]$$

$$\underset{d\times d}{\mathrm{var}\left[\hat{\boldsymbol{\theta}}(t)\right]} = \hat{\lambda}^2(t)\underset{d\times d}{P(t)}$$

The burst effect

If $\boldsymbol{\varphi}(t)$ is **zero** for some time, the parameter estimates will **remain constant**, since the algorithm gain $K(t) = P(t)\boldsymbol{\varphi}(t)$ is 0 and **no correction is applied**

At the same time, $S(t)$ will tend to **decrease** (if the forgetting factor $\mu < 1$):

$$\underset{d\times d}{S(t)} = \mu \cdot \underset{1\times 1}{S(t-1)} + \underset{d\times 1}{\boldsymbol{\varphi}(t)}\underset{1\times d}{\boldsymbol{\varphi}^{\mathsf{T}}(t)} = \mu \cdot S(t-1) \Rightarrow S(t) \to 0$$

Correspondingly, matrix $P(t)$ will **increase to very high values**

$$\underset{d\times d}{P(t)} = \frac{1}{\mu}\left[P(t-1) - \frac{P(t-1)\boldsymbol{\varphi}(t)\boldsymbol{\varphi}^{\mathsf{T}}(t)P(t-1)}{\mu + \boldsymbol{\varphi}^{\mathsf{T}}(t)P(t-1)\boldsymbol{\varphi}(t)}\right] = \frac{1}{\mu}P(t-1) \;\Rightarrow P(t) \to \infty$$

Now, if $\boldsymbol{\varphi}(t) \neq \mathbf{0}$ and $\varepsilon \neq 0$ at a subsequent time, then the parameter estimates will display an **abrupt variation (burst),** since $\widehat{\boldsymbol{\theta}}(t) = \widehat{\boldsymbol{\theta}}(t-1) + K(t)\cdot\varepsilon(t)$ and $K(t)$ **is very high**

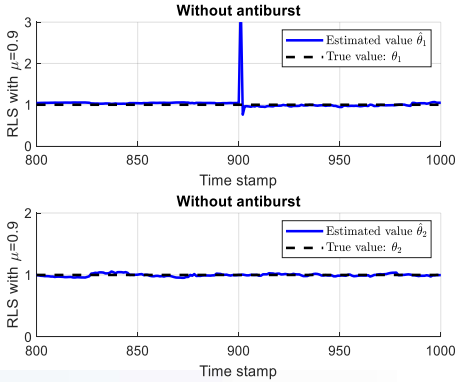
Example: the burst effect

Assume that the system generating the data is given as follows:

$S: y(t) = \theta_1(t)u_1(t) + \theta_2(t)u_2(t) + e(t)$

- $\theta_1(t) = \theta_2(t) = 1 \quad \forall t$
- $u_1(t) = \begin{cases} \sim \mathrm{WN}(0,1), & 1 \le t \le 100 \\ = 0, & 100 < t \le 900 \\ \sim \mathrm{WN}(0,1), & 900 < t \le 1000 \end{cases}$
- $u_2(t) = \mathrm{WN}(0,1)$
- $e(t) = \mathrm{WN}(0, 0.01)$

A burst phenomenon occurs just after sample 900, where the element $S(1,1)$ goes progressively to zero



Anti-burst remedies

One possible strategy consists in monitoring the algorithm for the wind-up of $K(t)$

For instance, one can monitor the **eigenvalues of** $P(t)$ and **reset** this matrix when they reach some upper bound. The gain is then evaluated with the resettled matrix

Another strategy consists in **keeping the trace of** $S(t)$ **constant**, thus indirectly avoiding the blow-up of the eigenvalues (*the trace equals the sum of the eigenvalues*). This is achieved by acting on the forgetting factor:

$$\underset{d\times d}{S(t)} = \mu\underset{1\times 1}{S(t-1)} + \underset{d\times 1}{\boldsymbol{\varphi}(t)}\underset{1\times d}{\boldsymbol{\varphi}^{\mathsf{T}}(t)}$$

$$\begin{aligned}\mathrm{tr}[S(t)] &= \mu \cdot \mathrm{tr}[S(t-1)] + \mathrm{tr}[\boldsymbol{\varphi}(t)\boldsymbol{\varphi}^{\mathsf{T}}(t)] \\ &= \mu \cdot \mathrm{tr}[S(t-1)] + \|\boldsymbol{\varphi}(t)\|^2\end{aligned}$$

The trace $\mathrm{tr}[S(t)]$ is **constant and equal to** k provided that $\mu(t) = 1 - \dfrac{\|\boldsymbol{\varphi}(t)\|^2}{k}$

With $\|\boldsymbol{\varphi}(t)\|$ we denote the Euclidean 2-norm

Anti-burst remedies

In fact, a constant trace of $S(t)$ means that we have that $\mathrm{tr}[S(t)] = \mathrm{tr}[S(t-1)] = k$

$\mathrm{tr}[S(t)] = \mu \cdot \mathrm{tr}[S(t-1)] + \|\boldsymbol{\varphi}(t)\|^2$ ⇨ Dividing by $\mathrm{tr}[S(t)]$ ⇨ $1 = \mu + \dfrac{1}{k}\|\boldsymbol{\varphi}(t)\|^2$

⇨ $\mu(t) = 1 - \dfrac{\|\boldsymbol{\varphi}(t)\|^2}{k}$

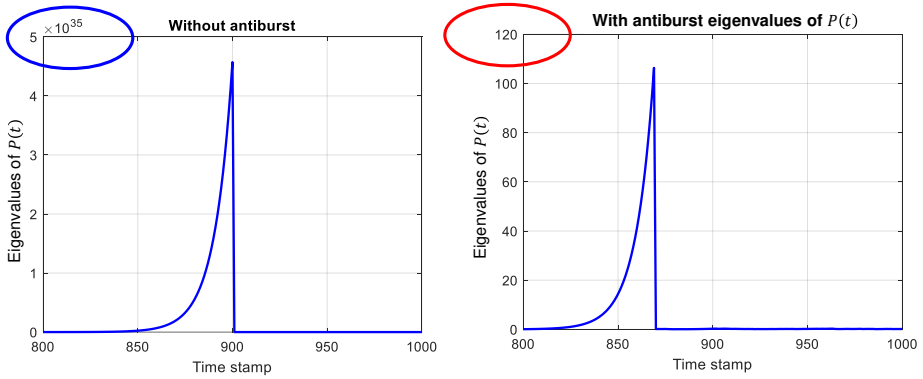
To avoid that $\mu(t)$ turns negative a **saturation** is also introduced:

$$\mu(t) = \max\left\{\mu_0, 1 - \frac{\|\boldsymbol{\varphi}(t)\|^2}{k}\right\}$$

μ_0 is a minimum value for $\mu(t)$ chosen a-priori

Example: the burst effect

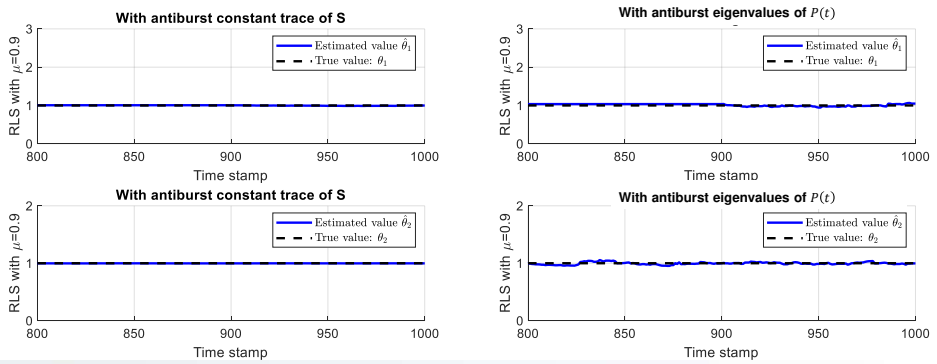
In the previous example, the eigenvalues of $P(t)$ start increasing after time step 100



Example: the burst effect

$k = 1000$ if we put a high enough k, it is improbable that some eigenvalue will be zero

Use Cholesky to find $P = LL^{\mathsf{T}}$, then substitute the highest value on $\mathrm{diag}(P)$ with a low value (e.g. 0.1) and then recompute P



Outline

1. Introduction to recursive and adaptive system identification
2. ARX models: Recursive Least Squares (RLS)
3. ARX models: adaptive identification and forgetting factor
4. **Connections with Kalman Filtering**
5. Least Mean Squares (LMS) algorithm

Connections with Kalman filtering

Advanced material

Assume that the parameters $\boldsymbol{\theta}$ are **constant**. The model

$$\underset{1\times 1}{y(t)} = \underset{1\times d}{\boldsymbol{\varphi}^{\mathsf{T}}(t)}\underset{d\times 1}{\boldsymbol{\theta}} + e(t) \qquad e(t) \sim \mathrm{WN}(0, \lambda^2)$$

can be described by the **state space** system

$$\begin{cases} \underset{d\times 1}{\boldsymbol{\theta}(t+1)} = \underset{d\times 1}{\boldsymbol{\theta}(t)} \\ \underset{1\times 1}{y(t)} = \underset{1\times d}{\boldsymbol{\varphi}^{\mathsf{T}}(t)}\underset{d\times 1}{\boldsymbol{\theta}} + e(t) \end{cases}$$

where the parameters vector $\boldsymbol{\theta}$ is interpreted as a **state vector**, where the state dimension n is equal to the parameters dimension d

A.H. Sayed, T. Kailath, A state space approach to adaptive RLS filtering, IEEE Signal Process. Mag. 11 (1994) 18–60.

Connections with Kalman filtering

Advanced material

If the parameters $\boldsymbol{\theta}(t)$ are **time-variant,** the model

$$y(t) = \underset{1\times 1}{\boldsymbol{\varphi}^{\mathsf{T}}(t)}\underset{d\times 1}{\boldsymbol{\theta}(t)} + e(t) \qquad e(t) \sim \mathrm{WN}(0, \lambda^2)$$

can be described by the **state space** system

$$\begin{cases} \underset{d\times 1}{\boldsymbol{\theta}(t+1)} = \underset{d\times 1}{\boldsymbol{\theta}(t)} + \underset{d\times 1}{\boldsymbol{v}_1(t)} & \boldsymbol{v}_1(t) \sim \mathrm{WN}(0, \underset{d\times d}{V_1}) \\ \underset{1\times 1}{y(t)} = \underset{1\times d}{\boldsymbol{\varphi}^{\mathsf{T}}(t)}\underset{d\times 1}{\boldsymbol{\theta}} + e(t) & \boldsymbol{v}_1(t) \perp e(t) \end{cases}$$

The **covariance matrix** V_1 can be used to describe **how fast** the different components of $\boldsymbol{\theta}(t)$ are **expected to vary**

Connections with Kalman filtering

Advanced material

Let's compare with a general time-variant state-space system

$$\begin{cases} \underset{d\times 1}{\boldsymbol{\theta}(t+1)} = \underset{d\times 1}{\boldsymbol{\theta}(t)} + \underset{d\times 1}{\boldsymbol{v}_1(t)} \\ \underset{1\times 1}{y(t)} = \underset{1\times d}{\boldsymbol{\varphi}^{\mathsf{T}}(t)}\underset{d\times 1}{\boldsymbol{\theta}} + e(t) \\ \boldsymbol{v}_1(t) \perp e(t) \end{cases} \qquad \begin{cases} \underset{n\times 1}{\boldsymbol{x}(t+1)} = A(t)\boldsymbol{x}(t) + \boldsymbol{v}_1(t) & \boldsymbol{v}_1(t) \sim \mathrm{WN}(0, \underset{n\times n}{V_1}) \\ \underset{p\times 1}{\boldsymbol{y}(t)} = \underset{p\times n}{C(t)}\underset{n\times 1}{\boldsymbol{x}(t)} + \boldsymbol{v}_2(t) & \boldsymbol{v}_2(t) \sim \mathrm{WN}(0, \underset{p\times p}{V_2}) \\ & \boldsymbol{v}_1(t) \perp \boldsymbol{v}_2(t) \end{cases}$$

The two systems are equivalent if $A(t) = I_d$, $C(t) = \boldsymbol{\varphi}^{\mathsf{T}}(t), \boldsymbol{v}_2(t) = e(t)$, $V_2 = \lambda^2$. Thus, the **Kalman filter recursions** can be applied to estimate the states (i.e. the parameters)

Connections with Kalman filtering

Advanced material

Kalman predictor:

$$\begin{cases} \underset{n\times 1}{\hat{\boldsymbol{x}}(t+1|t)} = \underset{n\times n}{A(t)}\underset{n\times 1}{\hat{\boldsymbol{x}}(t|t-1)} + \underset{n\times p}{K(t)}\underset{p\times 1}{\boldsymbol{\varepsilon}(t)} \\ \underset{p\times 1}{\hat{\boldsymbol{y}}(t|t-1)} = \underset{p\times n}{C(t)}\underset{n\times 1}{\hat{\boldsymbol{x}}(t|t-1)} \end{cases}$$

$$\begin{cases} \underset{p\times 1}{\boldsymbol{\varepsilon}(t)} = \underset{p\times 1}{\boldsymbol{y}(t)} - \underset{p\times 1}{\hat{\boldsymbol{y}}(t|t-1)} \\ \underset{n\times p}{K(t)} = \underset{n\times n}{A(t)}\underset{n\times n}{P^*(t)}\underset{n\times p}{C^{\mathsf{T}}(t)} \cdot \underset{p\times n}{(C(t)}\underset{n\times n}{P^*(t)}\underset{n\times p}{C^{\mathsf{T}}(t)} + \underset{p\times p}{V_2})^{-1} \\ \underset{n\times n}{P^*(t+1)} = \underset{n\times n}{A(t)}\underset{n\times n}{P^*(t)}\underset{n\times n}{A^{\mathsf{T}}(t)} - \underset{n\times p}{K(t)}\underset{p\times n}{C(t)}\underset{n\times n}{P^*(t)}\underset{n\times n}{A^{\mathsf{T}}(t)} + \underset{n\times n}{V_1} \end{cases}$$

Kalman filter:

$$\hat{\boldsymbol{x}}(t|t) = \hat{\boldsymbol{x}}(t|t-1) + K_0(t)\boldsymbol{\varepsilon}(t) \qquad \underset{n\times p}{K_0(t)} = \underset{n\times n}{P^*(t)}\underset{n\times p}{C^{\mathsf{T}}(t)}[\underset{p\times n}{C(t)}\underset{n\times n}{P^*(t)}\underset{n\times p}{C^{\mathsf{T}}(t)} + \underset{p\times p}{V_2}]^{-1}$$

Assumptions:

- $v_1(t) \perp v_2(t)$
- $v_1(t), v_2(t) \perp \boldsymbol{x}(0)$

Controlli automatici - 6 CFU

Connections with Kalman filtering

Advanced material

Kalman filter general form

Kalman filter:

$$\begin{cases} \hat{\boldsymbol{x}}(t|t-1) = A(t)\hat{\boldsymbol{x}}(t-1|t-1) + B\boldsymbol{u}(t) \\ \hat{\boldsymbol{x}}(t|t) = \hat{\boldsymbol{x}}(t|t-1) + K_0(t)\boldsymbol{\varepsilon}_y(t) \\ \hat{\boldsymbol{y}}(t|t-1) = C(t)\hat{\boldsymbol{x}}(t|t-1) + D\boldsymbol{u}(t) \end{cases}$$

$$\begin{cases} \boldsymbol{\varepsilon}_y(t) = \boldsymbol{y}(t) - \hat{\boldsymbol{y}}(t|t-1) \\ K_0(t) = P_0(t|t-1)C^{\mathsf{T}}(t)\mathrm{E}(t)^{-1} \\ \mathrm{E}(t) = C(t)P_0(t|t-1)C^{\mathsf{T}}(t) + V_2 \\ P_0(t|t-1) = A(t)P_0(t-1|t-1)A(t)^{\mathsf{T}} + V_1 \\ P_0(t|t) = \big(I_n - K_0(t)C(t)\big)P_0(t|t-1) \end{cases}$$

$$\hat{\boldsymbol{x}}(1|0) = \boldsymbol{x}^0$$
$$P_0(1|0) = P_0^0$$

$P_0(t|t)$ is the covariance matrix of the filtering estimate $\hat{\boldsymbol{x}}(t|t)$

Connections with Kalman filtering

Advanced material

Remarks:

- The connections with Kalman filtering allows to extend the RLS method also to **multiple output measurements**
- Initial values of $\widehat{\boldsymbol{\theta}}(0)$ and $P^*(0)$ assume the **interpretation of prior knowledge** on the parameters values and uncertainty, respectively
- Thus, if $P^*(0)$ is **large**, then the choice of $\widehat{\boldsymbol{\theta}}(0)$ has **little importance** and $\widehat{\boldsymbol{\theta}}(t)$ will **change fast**
- If **no prior information** is available, set $\widehat{\boldsymbol{\theta}}(0) = 0$, $P^*(0) = \rho I$ with ρ large

Connections with Kalman filtering

Advanced material

Remarks:

- The use of matrix V_1 allows to set **different changing rates** for the parameters, and **keeps $P^*(t)$ from going to zero** (so that the gain $K(t)$ never becomes zero)
- In the Kalman filter, the matrix $C(t)$ is supposed to **not depend on data**, which is not our case. However, provided that $\boldsymbol{\varphi}^{\mathsf{T}}(t)$ is free from measurement errors (i.e. **not an errors-in-variables situation**), the filter recursions are still valid

Outline

Least Mean Squares (LMS)

The RLS algorithm is **computationally demanding** due to the necessity to solve the auxiliary matrix recursion at each step.

A simpler scheme consists in simplifying the gain $K(t)$ to $\gamma \cdot \boldsymbol{\varphi}(t)$, where $\gamma > 0$ is a suitable small constant (denoted **step-size**)

In the RLS setting, we had $K(t) = P(t)\boldsymbol{\varphi}(t)$

The resulting algorithm is denoted **Least Mean Squares** (LMS):

LMS

- Basic recursion: $\underset{d\times 1}{\widehat{\boldsymbol{\theta}}(t)} = \underset{d\times 1}{\widehat{\boldsymbol{\theta}}(t-1)} + \underset{d\times 1}{K(t)} \cdot \underset{1\times 1}{\varepsilon(t)}$
- Definitions: $\underset{d\times 1}{K(t)} = \underset{1\times 1}{\gamma} \cdot \underset{d\times 1}{\boldsymbol{\varphi}(t)}$

  $\underset{1\times 1}{\varepsilon(t)} = y(t) - \boldsymbol{\varphi}^{\mathsf{T}}(t)\widehat{\boldsymbol{\theta}}(t-1)$

Least Mean Squares (LMS)

The rationale behind the LMS can be better understood by resorting to the classical **Newton's optimization method** for the minimization of

$$J(\boldsymbol{\theta}) = \sum_{t=1}^{N} \varepsilon(t)^2 = \sum_{t=1}^{N} (y(t) - \underset{1\times d}{\boldsymbol{\varphi}^{\top}(t)} \underset{d\times 1}{\boldsymbol{\theta}})^2$$

Newton's method is an **iterative method** that locally approximates the cost function with a **quadratic function** and takes the next point in the parameter space as the **minimum point of the quadratic** approximating function:

$$\underset{d\times 1}{\boldsymbol{\theta}^{(i+1)}} = \boldsymbol{\theta}^{(i)} - \left(\frac{d^2 J(\boldsymbol{\theta}^{(i)})}{d\boldsymbol{\theta}^{(i)2}}\right)^{-1} \cdot \frac{dJ(\boldsymbol{\theta}^{(i)})}{d\boldsymbol{\theta}^{(i)}}$$

Iteration $i+1$ — Iteration i — **Hessian** — **Gradient**

Least Mean Squares (LMS)

A simplified version of Newton's algorithm is the **steepest descent method**, that essentially approximates the Hessian with a constant, denoted α (and referred to as **convergence factor, step size or gain**):

$$\underset{d\times 1}{\boldsymbol{\theta}^{(i+1)}} = \boldsymbol{\theta}^{(i)} - \underset{1\times 1}{\frac{\alpha}{2}} \cdot \underset{d\times 1}{\frac{dJ(\boldsymbol{\theta}^{(i)})}{\partial\boldsymbol{\theta}^{(i)}}}$$

Gradient

The steepest descent method follows **strictly the negative gradient direction**. It is **generally slower** than Newton's method but requires **less computational effort**

Least Mean Squares (LMS)

Assuming that $\boldsymbol{\varphi}(t)$ depends only on measured data and it is independent of $\boldsymbol{\theta}$ (which occurs if the model has an ARX structure), then $\frac{\partial\varepsilon(t)}{\partial\boldsymbol{\theta}} = -\boldsymbol{\varphi}(t)$, and the gradient of $J(\cdot)$ is simply calculated as follows:

$$\underset{d\times 1}{\frac{\partial J(\boldsymbol{\theta})}{\partial\boldsymbol{\theta}}} = -2\sum_{t=1}^{N} \underset{d\times 1}{\boldsymbol{\varphi}(t)} \underset{1\times 1}{\varepsilon(t)}$$

Computing this quantity is **impossible in adaptive applications**, since data are available only up to time t. For this reason, a different and simpler cost function is minimized, namely the **instantaneous square error** (*stochastic gradient approximation*):

$$\hat{J}(\boldsymbol{\theta}) = \varepsilon^2(t) = (y(t) - \boldsymbol{\varphi}^{\top}(t)\boldsymbol{\theta})^2$$

Least Mean Squares (LMS)

$$\hat{J}(\boldsymbol{\theta}) = \varepsilon^2(t) = (y(t) - \boldsymbol{\varphi}^{\top}(t)\boldsymbol{\theta})^2$$

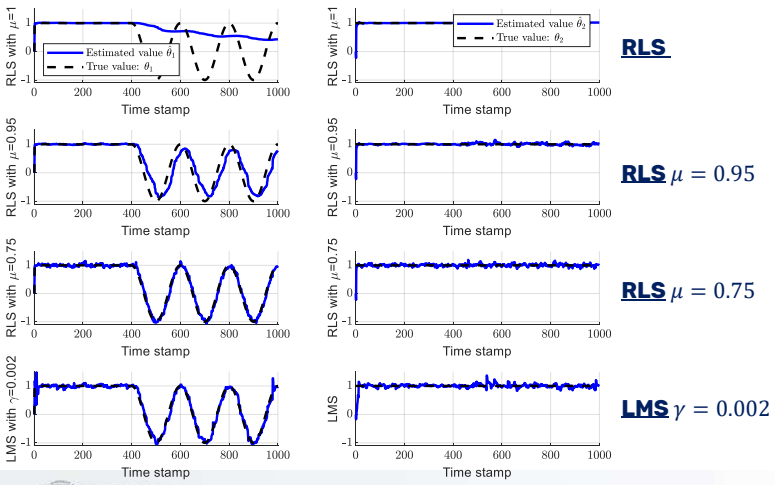
We then have:

$$\underset{d\times 1}{\frac{\partial\hat{J}(\boldsymbol{\theta})}{\partial\boldsymbol{\theta}}} = -2\underset{d\times 1}{\boldsymbol{\varphi}(t)}\varepsilon(t)$$

which leads to the following update rule:

$$\underset{d\times 1}{\boldsymbol{\theta}^{(i+1)}} = \boldsymbol{\theta}^{(i)} + \underset{1\times 1}{\alpha} \cdot \underset{d\times 1}{\boldsymbol{\varphi}(t)} \underset{1\times 1}{\varepsilon(t)}$$

which coincides with the LMS rule if $\alpha = \gamma$ and if we **synchronize iterations** i **and sampling times** t (i.e. if $t = i$)



RLS

RLS $\mu = 0.95$

RLS $\mu = 0.75$

LMS $\gamma = 0.002$

Least Mean Squares (LMS)

Remarks:

- Since the **gain** γ is **fixed** and **never zero**, the algorithm is **always «alert»** to track parameters change
- LMS is particularly suitable for **adaptive applications**, since the **gain does not suffer from build-up problems**
- LMS is **computationally less intensive** than RLS, since it avoids the auxiliary recursion. For both these reasons, LMS has had a tremendous impact in the field of adaptive signal processing

Least Mean Squares

Remarks:

- LMS employs the information conveyed by the data in an **approximate way** *(stochastic gradient descent):* the sample $\boldsymbol{\varphi}(t)$ is used only at time t, and does not influence the future behaviour of the estimates
- The pure LMS is sensitive to the scaling of $\boldsymbol{\varphi}(t)$. This makes it difficult to choose the step-size γ. A **normalized version (NLMS)** sets the gain to

$$K(t) = \gamma \cdot \frac{\boldsymbol{\varphi}(t)}{1 + \|\boldsymbol{\varphi}(t)\|^2}$$

UNIVERSITÀ DEGLI STUDI DI BERGAMO | Dipartimento di Ingegneria Gestionale, dell'Informazione e della Produzione

UNIVERSITÀ DEGLI STUDI DI BERGAMO | Dipartimento di Ingegneria Gestionale, dell'Informazione e della Produzione

ADAPTIVE LEARNING, ESTIMATION AND SUPERVISION OF DYNAMICAL SYSTEMS (ALES)

Lecture 3: Instrumental variables

Master Degree in COMPUTER ENGINEERING

Data Science and Data Engineering Curriculum

SPEAKER
Prof. Mirko Mazzoleni

PLACE
University of Bergamo

Outline

Outline

1. Least squares revised
2. The instrumental variable (IV) method
3. ARX model estimation with instrumental variables
4. Recursive instrumental variable method

Outline

1. **Least squares revised**
2. The instrumental variable (IV) method
3. ARX model estimation with instrumental variables
4. Recursive instrumental variable method

Least squares revised

Let's analyze the **correctness** of the solution provided by **least squares**. Consider a **static** setting, and assume that the data have been generated by the linear system

$$\mathcal{S}: \; y(i) = \underset{1\times d}{\boldsymbol{\varphi}^\top(i)} \underset{d\times 1}{\boldsymbol{\theta}^0} + e(i) \qquad e(i) \sim \mathrm{WN}(0, \lambda_0^2)$$

Modeling the data with a **linear model**

$$\mathcal{M}: \; y(i) = \underset{1\times d}{\boldsymbol{\varphi}^\top(i)} \underset{d\times 1}{\boldsymbol{\theta}} + \eta(i) \qquad \eta(i) \sim \mathrm{WN}(0, \lambda^2)$$

and using a **least squares** estimation criterion, gives the solution

$$\underset{d\times 1}{\widehat{\boldsymbol{\theta}}_{\mathrm{LS}}} = \left(\sum_{i=1}^{N} \underset{d\times 1}{\boldsymbol{\varphi}(i)} \underset{1\times d}{\boldsymbol{\varphi}^\top(i)}\right)^{-1} \cdot \sum_{i=1}^{N} \underset{d\times 1}{\boldsymbol{\varphi}(i)} \underset{1\times 1}{y(i)}$$

Least squares revised

Focus now on the **estimation error** $\widehat{\boldsymbol{\theta}}_{\mathrm{LS}} - \boldsymbol{\theta}^0$

$$\widehat{\boldsymbol{\theta}}_{\mathrm{LS}} - \boldsymbol{\theta}^0 = \left(\sum_{i=1}^{N} \boldsymbol{\varphi}(i)\boldsymbol{\varphi}^\top(i)\right)^{-1} \sum_{i=1}^{N} \boldsymbol{\varphi}(i) y(i) - \boldsymbol{\theta}^0$$

$$= \left(\sum_{i=1}^{N} \boldsymbol{\varphi}(i)\boldsymbol{\varphi}^\top(i)\right)^{-1} \cdot \left[\sum_{i=1}^{N} \boldsymbol{\varphi}(i) y(i) - \left\{\sum_{i=1}^{N} \boldsymbol{\varphi}(i)\boldsymbol{\varphi}^\top(i)\right\} \boldsymbol{\theta}^0\right]$$

$$= \left(\sum_{i=1}^{N} \boldsymbol{\varphi}(i)\boldsymbol{\varphi}^\top(i)\right)^{-1} \cdot \left[\sum_{i=1}^{N} \boldsymbol{\varphi}(i)\{y(i) - \boldsymbol{\varphi}^\top(i)\boldsymbol{\theta}^0\}\right] = \left(\sum_{i=1}^{N} \boldsymbol{\varphi}(i)\boldsymbol{\varphi}^\top(i)\right)^{-1} \cdot \sum_{i=1}^{N} \boldsymbol{\varphi}(i) e(i)$$

Least squares revised

$$\widehat{\boldsymbol{\theta}}_{\mathrm{LS}} - \boldsymbol{\theta}^0 = \left(\sum_{i=1}^{N} \boldsymbol{\varphi}(i)\boldsymbol{\varphi}(i)^\top\right)^{-1} \cdot \sum_{i=1}^{N} \boldsymbol{\varphi}(i) e(i)$$

The summations in the equation **converge to their expected values** as $N \to +\infty$, under mild conditions. Thus, the estimate $\widehat{\boldsymbol{\theta}}_{\mathrm{LS}}$ is **consistent** if:

- $\mathbb{E}[\boldsymbol{\varphi}(i)\boldsymbol{\varphi}(i)^\top]$ **is invertible**
- $\mathbb{E}[\boldsymbol{\varphi}(i)e(i)] = \mathbf{0}$ ⇨ **Not always true!**

Example: least squares with noise on regressors

Assume the following data-generating mechanism:

$y(i) = \theta x_0(i) + e_y(i), \quad \theta = 3, \quad e_y \sim \mathrm{WN}(0,4)$

$x_0 \sim \mathrm{WN}(0,1)$

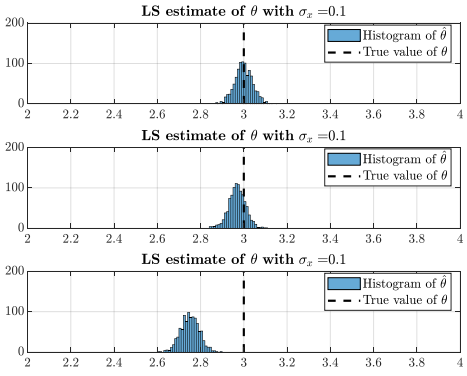
so that $x_0(i)$ is **noiseless**.

Suppose now that we measure $N = 2500$ data and employ

$$x(i) = x_0(i) + e_x(i) \qquad e_x \sim \mathrm{WN}(0, \sigma_x^2)$$

instead of $x_0(i)$, i.e. we have a **noisy regressor** $x(i)$.

The **LS estimates are biased** proportionally to σ_x^2



Example: least squares with noise on regressors

Let's investigate the problem. The relation assumed by least squares is

$$y(i) = \theta x_0(i) + e_y(i), \quad \theta = 3, \quad e_y \sim \mathrm{GWN}(0, \sigma_y^2)$$

but we employ $x(i) = x_0(i) + e_x(i)$ in place of $x_0(i)$

$$\Downarrow$$

$$y(i) = \theta[x(i) - e_x(i)] + e_y(i) = \theta x(i) \overbrace{- \theta e_x(i) + e_y(i)}^{e(i)} = \theta x(i) + e(i)$$

So that the employed regressor x **is correlated with the output noise** $e(\cdot)$:

$$\mathbb{E}[x(i) \cdot e(i)] = \mathbb{E}\left[\left(x_0(i) + e_x(i)\right) \cdot \left(-\theta e_x(i) + e_y(i)\right)\right] \neq 0$$

So, a **bias is present in the estimate** of θ

Outline

The instrumental variable method

Instrumental variable methods can be seen as **generalizations** of the **least squares** estimates. The main idea is to modify the LS estimate so that it is **consistent** for an **arbitrary disturbance**

Consider again a static linear model estimation. The normal equations are modified as

$$\widehat{\boldsymbol{\theta}}_{\mathrm{IV}} = \left(\sum_{i=1}^{N} \underset{d\times 1}{\boldsymbol{z}(i)} \underset{1\times d}{\boldsymbol{\varphi}(i)^{\top}}\right)^{-1} \sum_{i=1}^{N} \boldsymbol{z}(i) y(i)$$

where $\boldsymbol{z}(t) \in \mathbb{R}^{d\times 1}$ is a vector called **instrumental variable**, that must satisfy certain conditions to guarantee the consistency of the estimates $\widehat{\boldsymbol{\theta}}_{\mathrm{IV}}$. Notice that for $\boldsymbol{z}(t) = \boldsymbol{\varphi}(t)$ we have that $\widehat{\boldsymbol{\theta}}_{\mathrm{IV}} = \widehat{\boldsymbol{\theta}}_{\mathrm{LS}}$

The instrumental variable method

Assuming again a linear data-generating system, we want that:

- $\mathbb{E}\left[\boldsymbol{z}(i)\boldsymbol{\varphi}(i)^{\top}\right]$ **is invertible**
- $\mathbb{E}[\boldsymbol{z}(i)e(i)] = \mathbf{0}$

so that the **bias in the estimate is eliminated**

With these assumptions, by analyzing the estimation error $\widehat{\boldsymbol{\theta}}_{\mathrm{IV}} - \boldsymbol{\theta}^0$ in a similar way as before, we have that

$$\widehat{\boldsymbol{\theta}}_{\mathrm{IV}} - \boldsymbol{\theta}^0 = \left(\sum_{i=1}^{N} \boldsymbol{z}(i)\boldsymbol{\varphi}(i)^{\top}\right)^{-1} \cdot \sum_{i=1}^{N} \boldsymbol{z}(i)e(i) = \mathbf{0}$$

Outline

1. Least squares revised
2. The instrumental variables (IV) method
3. **ARX model estimation with instrumental variables**
4. Recursive instrumental variable method
5. Application study: Virtual Reference Feedback Tuning (VRFT)

ARX model estimation with instrumental variables

Suppose that the data are generated by an ARMAX system, so that

$$\mathcal{S}:\ y(t) = \frac{B_0(z)}{A_0(z)}u(t-1) + \frac{C_0(z)}{A_0(z)}e(t) \qquad e(t) \sim \mathrm{WN}(0, \lambda^2)$$

In these cases, the statistically optimal solution is the **Maximum Likelihood (ML)** approach *(see IMAD course)*

However, the optimization scheme is **iterative** and suffers from **local minima**. Furthermore, the **Recursive ML** approach is possible but **computationally heavy**

If the **noise model is not of interest**, estimating an ARX model is simpler and more computationally efficient

ARX model estimation with instrumental variables

Problem formulation:

We want to estimate an **ARX model**

$$\mathcal{M}:\ y(t) = \frac{B(z;\boldsymbol{\theta})}{A(z;\boldsymbol{\theta})}u(t-1) + \frac{1}{A(z;\boldsymbol{\theta})}\eta(t) \qquad \eta(t) \sim \mathrm{WN}(0, \mu^2)$$

when the data are generated from an **ARMAX system** ($C(z) \neq 1$)

$$\mathcal{S}:\ y(t) = \frac{B_0(z;\boldsymbol{\theta}^0)}{A_0(z;\boldsymbol{\theta}^0)}u(t-1) + \frac{C_0(z;\boldsymbol{\theta}^0, c_1, \dots, c_{n_c})}{A_0(z;\boldsymbol{\theta}^0)}e(t) \qquad e(t) \sim \mathrm{WN}(0, \lambda^2)$$

$$\boldsymbol{\theta}^0 = \begin{bmatrix} a_1^0 & \dots & a_{n_a}^0 & b_0^0 & \dots & b_{n_b}^0 \end{bmatrix}^\top$$

The estimation problem is therefore to **estimate the set of true parameters** $\boldsymbol{\theta}^0$, defined as the polynomial coefficients of $A(z)$ and $B(z)$, **neglecting the estimation** of $C(z)$

ARX model estimation with instrumental variables

Hypothesis: $\dfrac{B_0(z;\boldsymbol{\theta}^0)}{A_0(z;\boldsymbol{\theta}^0)} \in \dfrac{B(z;\boldsymbol{\theta})}{A(z;\boldsymbol{\theta})}$

The problem depends on these facts:

- we want to **identify ARX models** (for simplicity)
- we want to estimate $\dfrac{B_0(z;\boldsymbol{\theta}^0)}{A_0(z;\boldsymbol{\theta}^0)}$ **exactly**
- $\dfrac{B_0(z;\boldsymbol{\theta}^0)}{A_0(z;\boldsymbol{\theta}^0)} \in \dfrac{B(z;\boldsymbol{\theta})}{A(z;\boldsymbol{\theta})}$ **does not grant** an exact estimate of this part

ARX model estimation with instrumental variables

ARX model in linear regression form:

$$y(t) = \boldsymbol{\varphi}(t)^\top \boldsymbol{\theta} + \eta(t)$$

$$\boldsymbol{\varphi}(t) = [y(t-1) \quad \dots \quad y(t-n_a) \quad u(t-1) \quad \dots \quad u(t-n_b-1)]^\top$$

$$\boldsymbol{\theta} = [a_1 \quad \dots \quad a_{n_a} \quad b_0 \quad \dots \quad b_{n_b}]^\top$$

ARMAX system in linear regression form:

$$y(t) = a_1^0 y(t-1) + \cdots + a_{n_a}^0 y(t-n_a) + b_0^0 u(t-1) + \cdots + b_{n_b}^0 u(t-n_b-1) + e(t) + \cdots + c_{n_c}^0 e(t-n_c)$$

$$y(t) = \boldsymbol{\varphi}(t)^\top \boldsymbol{\theta}_0 + v(t),$$

$$\boldsymbol{\theta}_0 = [a_1^0 \quad \dots \quad a_{n_a}^0 \quad b_0^0 \quad \dots \quad b_{n_b}^0]^\top$$

$$v(t) = e(t) + c_1^0 e(t-1) + \cdots + c_{n_c}^0 e(t-n_c)$$

$\Downarrow$

$v(t)$ **is correlated with** $\boldsymbol{\varphi}(t)$, unless $C_0(z) = 1$

ARX model estimation with instrumental variables

Since the **regressors** $\boldsymbol{\varphi}(t)$ **are correlated with the output noise** $v(t)$, a standard least squares approach will get biased estimates

Thus, the instrumental variable method can be applied. How can we design the instruments in this case? There are two (of may) solutions:

- **«double experiment»** method
- **«two-steps experiment»** method

Double experiment method

The method runs two experiments with the **same input** and then collect the outputs

1° experiment $\rightarrow \{u(1), u(2), \dots, u(N)\}, \ \{y(1), y(2), \dots, y(N)\}$

$$y(t) = \frac{B_0(z;\boldsymbol{\theta})}{A_0(z;\boldsymbol{\theta})} u(t-1) + \frac{C_0(z;\boldsymbol{\theta})}{A_0(z;\boldsymbol{\theta})} e(t) \qquad e(t) \sim \mathrm{WN}(0, \lambda^2)$$

2° experiment $\rightarrow \{u(1), u(2), \dots, u(N)\}, \ \{y'(1), y'(2), \dots, y'(N)\}$

$$y'(t) = \frac{B_0(z;\boldsymbol{\theta})}{A_0(z;\boldsymbol{\theta})} u(t-1) + \frac{C_0(z;\boldsymbol{\theta})}{A_0(z;\boldsymbol{\theta})} e'(t) \qquad e'(t) \sim \mathrm{WN}(0, \lambda^2)$$

Remark:

In the two experiments, only the **inputs are the same**; on the contrary, **the outputs are different**, due to a different noise realization

Double experiment method

We can design an IV as:

$$\boldsymbol{z}(t) = \begin{bmatrix} y'(t-1) \\ y'(t-2) \\ \vdots \\ y'(t-n_a) \\ u(t-1) \\ u(t-2) \\ \\ u(t-n_b-1) \end{bmatrix}$$

- $\boldsymbol{z}(t)$ and $\boldsymbol{\varphi}(t)$ are **strongly correlated**, just differing in the noise realization
- $\boldsymbol{z}(t) \perp e_0(t)$ assuming:
  - ✓ $e(t) \perp e'(t) \rightarrow$ true if unpredictable **white noise**
  - ✓ $u(t) \perp e(t) \rightarrow$ true in case of **open-loop** systems

- **Advantage:** we used two experiments with exactly corresponding inputs (signal with the greatest interest for constructing the IV)
- **Disadvantage:** it is not always possible to make a double experiment, i.e. in case of safety-critical systems

Two-steps method

The method builds a new signal $\check{y}(t)$ so that it is the **output of a deterministic system**

$$\check{y}(t) = \frac{\check{B}(z)}{\check{A}(z)} u(t-1)$$

where $\frac{\check{B}(z)}{\check{A}(z)}$ is an asymptotically stable filter and $u(t)$ is the **measured input**. Then we set

$$\boldsymbol{z}(t) = \begin{bmatrix} \check{y}(t-1) \\ \check{y}(t-2) \\ \vdots \\ \check{y}(t-n_a) \\ u(t-1) \\ u(t-2) \\ \\ u(t-n_b-1) \end{bmatrix}$$

- The $u(\cdot)$ part is **greatly correlated**
- $\boldsymbol{z}(t) \perp v(t)$ if $e(t) \perp u(t)$, since $\check{y}(t)$ is **noiseless**

Two-steps method

- **Advantage:** no double experiment is requested
- **Disadvantage:** many degrees of freedom in the $\frac{\check{B}(z)}{\check{A}(z)}$ filter choice

Choice of the filter

Simplest option: set $\frac{\check{B}(z)}{\check{A}(z)} = 1 \Rightarrow \check{y}(t) = u(t-1)$

Alternative option:

1) use a PEM method to estimate $\frac{\hat{B}(z;\hat{\boldsymbol{\theta}})}{\hat{A}(z;\hat{\boldsymbol{\theta}})}$ from $\{u(t), y(t)\}_{t=1}^{N}$ (which will be an incorrect model but somewhat close to $\frac{B_0(z;\boldsymbol{\theta}_0)}{A_0(z;\boldsymbol{\theta}_0)}$)
2) Get $\check{y}(t) = \frac{\hat{B}(z;\hat{\boldsymbol{\theta}})}{\hat{A}(z;\hat{\boldsymbol{\theta}})} u(t-1)$

Example: ARX model from ARMAX generated data

Assume that the system generating the data is an $\mathrm{ARMAX}(n_a = 2, n_c = 2, n_b = 0, k = 1)$

$$\mathcal{S}\colon\ y(t) = a_1^0 y(t-1) + a_2^0 y(t-2) + e(t) + c_1^0 e(t-1) + c_2^0 e(t-2) + b_1 u(t-1)$$

$$e(\cdot) = u(\cdot) = \mathrm{WN}(0,1), \qquad e(\cdot) \perp u(\cdot)$$

$$\boldsymbol{\theta}^0 = [a_1^0 \ \ a_2^0 \ \ b_1^0 \ c_1^0 \ \ c_2^0]^\top = [0.2 \ \ -0.5 \ \ 1 \ \ 0.5 \ \ 0.75]^\top$$

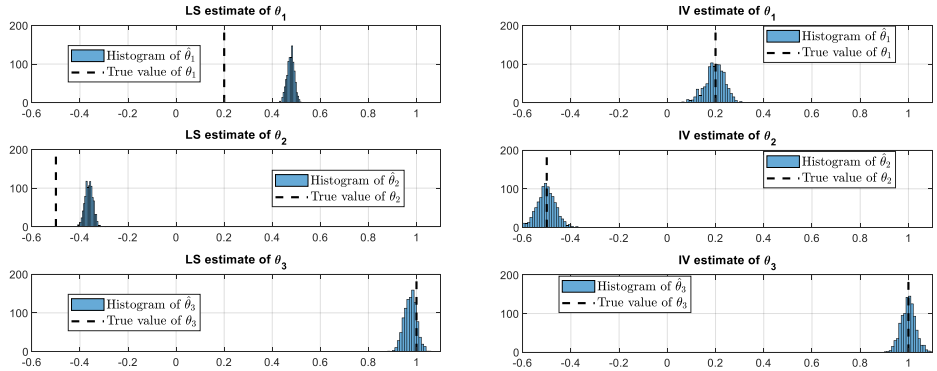
We want to estimate an $\mathrm{ARX}(n_a = 2, n_b = 0, k = 1)$ model

$$\mathcal{M}\colon\ y(t) = a_1 y(t-1) + a_2 y(t-2) + b_1 u(t-1) + \eta(t), \quad \eta(\cdot) \sim \mathrm{WN}(0, \lambda^2) \quad u(\cdot) = \mathrm{WN}(0,1)$$

In order to obtain **consistent estimates**, we employ the **IV approach** with the **double experiment** method. We collect $N = 2500$ data and perfom 1000 Monte Carlo runs

Example: ARX model from ARMAX generated data

Least squares | **Instrumental variables**



Instrumental variable identification schemes

The presented IV approach is employed for estimating the $A(z)$ and $B(z)$ polynomials of an ARX model

Alternative IV approaces, such as the **Refined Instrumental Variable (RIV)** method, allows to estimate also the noise model in an optimal way, as an alternative to the PEM/ML approach. This scheme can be applied to both continuous and discrete-time systems

Outline

1. Least squares revised
2. The instrumental variables (IV) method
3. ARX model estimation with instrumental variables
4. **Recursive instrumental variable method**

Recursive instrumental variable method

The instrumental variable batch solution can be computed recursively, just as for the RLS

Recursive IV summary

- Basic recursion: $\underset{d\times 1}{\hat{\boldsymbol{\theta}}(t)} = \hat{\boldsymbol{\theta}}(t-1) + K(t)\cdot\varepsilon(t)$
- Definitions: $\underset{d\times 1}{K(t)} = \underset{d\times d}{P(t)}\underset{d\times 1}{\boldsymbol{z}(t)} = \dfrac{P(t-1)\boldsymbol{z}(t)}{1+\boldsymbol{\varphi}^{\mathsf{T}}(t)P(t-1)\boldsymbol{z}(t)}$

  $\underset{1\times 1}{\varepsilon(t)} = y(t) - \boldsymbol{\varphi}^{\mathsf{T}}(t)\hat{\boldsymbol{\theta}}(t-1)$
- Auxiliary recursion: $\underset{d\times d}{P(t)} = P(t-1) - \beta(t-1)^{-1}\cdot P(t-1)\boldsymbol{z}(t)\boldsymbol{\varphi}^{\mathsf{T}}(t)P(t-1)$

  $\underset{1\times 1}{\beta(t-1)} = 1 + \underset{1\times d}{\boldsymbol{\varphi}^{\mathsf{T}}(t)}P(t-1)\underset{d\times 1}{\boldsymbol{z}(t)}$

Recursive instrumental variable method

For this recursive IV algorithm, an often employed approach is to use the current estimate of $\hat{\boldsymbol{\theta}}(t)$ as parameters for $\hat{B}(z;\hat{\boldsymbol{\theta}})/\hat{A}(z;\hat{\boldsymbol{\theta}})$ so that the instrumental variable using the two-steps method

$$\boldsymbol{z}(t) = [\check{y}(t-1) \quad \dots \quad \check{y}(t-n_a) \quad u(t-1) \quad \dots \quad u(t-n_b-1)]^{\mathsf{T}}$$

is built, at each time stamp, by computing $\check{y}(\cdot)$ as

$$\check{y}(t) = \boldsymbol{z}^{\mathsf{T}}(t)\hat{\boldsymbol{\theta}}(t)$$

where $\hat{\boldsymbol{\theta}} = [\hat{a}_1 \quad \dots \quad \hat{a}_{n_a} \quad \hat{b}_0 \quad \dots \quad \hat{b}_{n_b}]^{\mathsf{T}}$

UNIVERSITÀ DEGLI STUDI DI BERGAMO | Dipartimento di Ingegneria Gestionale, dell'Informazione e della Produzione

UNIVERSITÀ DEGLI STUDI DI BERGAMO | Dipartimento di Ingegneria Gestionale, dell'Informazione e della Produzione

ADAPTIVE LEARNING, ESTIMATION AND SUPERVISION OF DYNAMICAL SYSTEMS (ALES)

Lecture 4: Closed-loop identification

Master Degree in COMPUTER ENGINEERING

Data Science and Data Engineering Curriculum

SPEAKER
Prof. Mirko Mazzoleni

PLACE
University of Bergamo

Syllabus

1. **Recursive and adaptive identification**
   - 1.1 Recursive ARX estimation (RLS)
   - 1.2 Least Mean Squares (LMS)
   - 1.3 Instrumental Variables (IV)
2. **Closed-loop identification**
3. **Subspace and MIMO identification**
   - 3.1 Singular Value Decomposition
   - 3.2 Impulse data: Ho-Kalman, Kung algorithms
   - 3.3 Generic I/O data: the MOESP algorithm
4. **Supervision of dynamical systems**
   - 4.1 Introduction to fault diagnosis
   - 4.2 Model-based fault diagnosis
   - 4.3 Parity space approaches
   - 4.4 Observer-based approaches
   - 4.5 Signal-based fault diagnosis
   - 4.6 Knowledge-based fault diagnosis

CASE STUDIES

- Virtual Reference Feedback Tuning
- Nuclear particles classification
- Leak detection in an industrial valve
- Bearing fault identification

Outline

1. Introduction to closed-loop identification
2. Instrumental variables method
3. Direct identification method
4. Indirect identification methods

Outline

1. **Introduction to closed-loop identification**
2. Instrumental variables method
3. Direct identification method
4. Indirect identification methods

Introduction to closed-loop identification

Many systems operate under **feedback control**, due to:

- **safety** of operations
- **unstable behaviour** of open-loop plant

As a consequence, **experiments** can only be performed under presence of a **stabilizing controller**

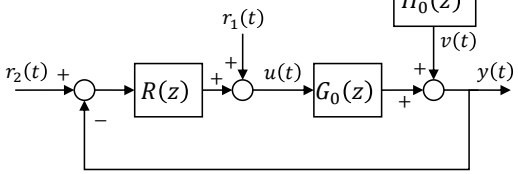
Another reason for performing closed-loop experiments is when we want to **improve an existing controller**

In this case, the dynamics that the plant exhibits under the old controller can be more relevant than its open-loop dynamics

Introduction to closed-loop identification

We consider **SISO** systems operating in **closed-loop** modeled as follows

$$\begin{cases} u(t) = R(z)[r_2(t) - y(t)] + r_1(t) \\ y(t) = G_0(z)u(t) + H_0(z)e(t) \end{cases}$$



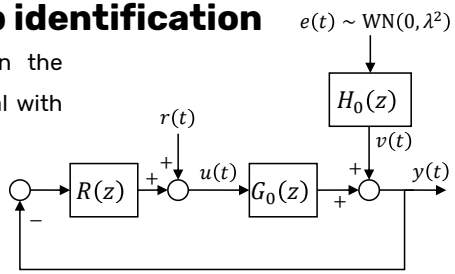
- $r_1(t)$ can be a **reference** value, a **setpoint** or a **noise** on the regulator output
- $r_2(t)$ can be a **setpoint** or a measurement **noise** on the output signal

Main difference w.r.t. the open-loop case: $u(t)$ and $e(t)$ are **correlated!**

Introduction to closed-loop identification

In situations that we are not interested in the separate effects of $r_1(t)$ and $r_2(t)$ we will deal with the signal $r(t)$ defined by

$$r(t) \coloneqq r_1(t) + R(z)r_2(t)$$



The system's equations become

$$\begin{cases} u(t) = r(t) - R(z)y(t) \\ y(t) = G_0(z)u(t) + v(t) \end{cases}$$

Introduction to closed-loop identification

$$\begin{cases} u(t) = r(t) - R(z)y(t) \\ y(t) = G_0(z)u(t) + v(t) \end{cases} \Rightarrow \begin{cases} u(t) = r(t) - R(z)y(t) \\ y(t) = G_0(z)r(t) - G_0(z)R(z)y(t) + v(t) \end{cases}$$

$$\begin{cases} u(t) = r(t) - R(z)y(t) \\ y(t) = \dfrac{G_0(z)}{1+G_0(z)R(z)}r(t) + \dfrac{1}{1+G_0(z)R(z)}v(t) \end{cases} \Rightarrow \begin{cases} u(t) = r(t) - R(z)\left[\dfrac{G_0(z)}{1+G_0(z)R(z)}r(t) + \dfrac{1}{1+G_0(z)R(z)}v(t)\right] \\ y(t) = \dfrac{G_0(z)}{1+G_0(z)R(z)}r(t) + \dfrac{1}{1+G_0(z)R(z)}v(t) \end{cases}$$

$$\begin{cases} u(t) = \dfrac{1}{1+G_0(z)R(z)}r(t) - \dfrac{R(z)}{1+G_0(z)R(z)}v(t) \\ y(t) = \dfrac{G_0(z)}{1+G_0(z)R(z)}r(t) + \dfrac{1}{1+G_0(z)R(z)}v(t) \end{cases}$$

Introduction to closed-loop identification

$$\begin{cases} u(t) = \dfrac{1}{1+G_0(z)R(z)}r(t) - \dfrac{R(z)}{1+G_0(z)R(z)}v(t) \\ y(t) = \dfrac{G_0(z)}{1+G_0(z)R(z)}r(t) + \dfrac{1}{1+G_0(z)R(z)}v(t) \end{cases}$$

Using the definition of the **sensitivity function** $S_0(z) := \frac{1}{1+G_0(z)R(z)}$, we have

$$\begin{cases} u(t) = S_0(z)r(t) - R(z)S_0(z)v(t) \\ y(t) = G_0(z)S_0(z)r(t) + S_0(z)v(t) \end{cases}$$

Is there a closed-loop identification problem?

Example: nonparametric spectral analysis

IMAD lecture 12

Consider a nonparametric spectral analysis/ETFE estimate using $u(t)$ and $y(t)$ only

$$\begin{cases} u(t) = S_0(z)[r(t) - R(z)v(t)] \\ y(t) = S_0(z)[G_0(z)r(t) + v(t)] \end{cases} \qquad \hat{G}(e^{j\omega}) = \frac{\hat{\Gamma}_{uy}(\omega)}{\hat{\Gamma}_{uu}(\omega)}$$

It can be shown that

$$\hat{G}(e^{j\omega}) = \frac{G_0(e^{j\omega})\Gamma_{rr}(\omega) - R^*(e^{j\omega})\Gamma_{vv}(\omega)}{\Gamma_{rr}(\omega) + |R(e^{j\omega})|^2\Gamma_{vv}(\omega)}$$

Is there a closed-loop identification problem?

$$\hat{G}(e^{j\omega}) = \frac{G_0(e^{j\omega})\Gamma_{rr}(\omega) - R^*(e^{j\omega})\Gamma_{vv}(\omega)}{\Gamma_{rr}(\omega) + |R(e^{j\omega})|^2\Gamma_{vv}(\omega)}$$

Limit cases:

- $\Gamma_{vv}(\omega) = 0$ **no noise** $\Rightarrow$ $\hat{G}(e^{j\omega}) = G_0(e^{j\omega})$
- $\Gamma_{rr}(\omega) = 0$ **no excitation** $\Rightarrow$ $\hat{G}(e^{j\omega}) = -\frac{1}{R(e^{j\omega})}$

When both noise and excitation are present (typical situation) the nonparametric **estimate** will be a **linear combination** of the two components (depending on the SNR)

Is there a closed-loop identification problem?

Consider the $\mathrm{OE}(n_b = 1, n_a = 4, k = 3)$ system sampled at $T_s = 0.05$ s with $e(t) \sim \mathrm{WN}(0, \lambda^2)$

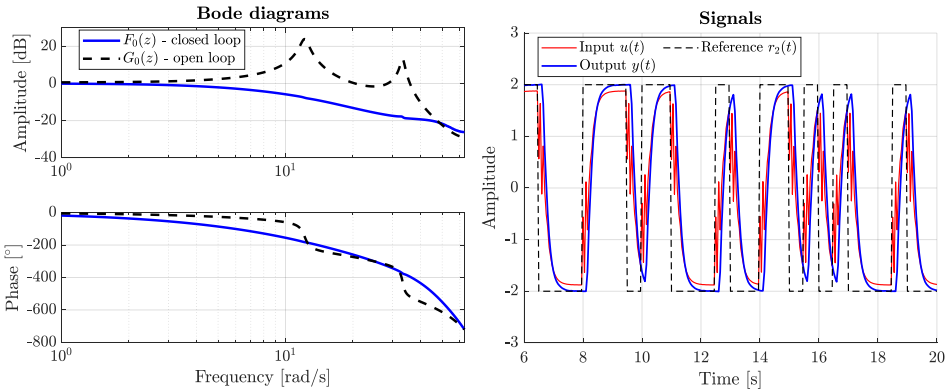
$$\mathcal{S}\colon\ y(t) = \frac{(0.28261 + 0.50666z^{-1}) \cdot z^{-3}}{1 - 1.41833z^{-1} + 1.58939z^{-2} - 1.31608z^{-3} + 0.88642z^{-4}} u(t) + e(t)$$

controlled with the regulator

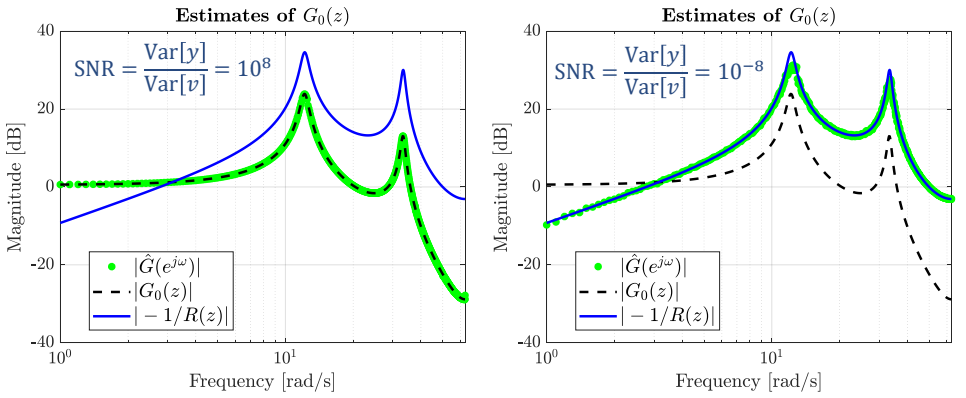
$$R(z) = \frac{0.32905 - 0.59771z^{-1} + 0.70728z^{-2} - 0.64010z^{-3} + 0.46499z^{-4} - 0.11769z^{-5}}{1 - z^{-1}}$$

Let $r_2(t)$ be a PRBS with levels $[-2, 2]$ and frequency content up to $0.1 \cdot \frac{f_s}{2} = 1$ Hz and $r_1(t) = 0$. Collect $N = 10000$ data in closed-loop. Use the **spafdr** Matlab command

Is there a closed-loop identification problem?

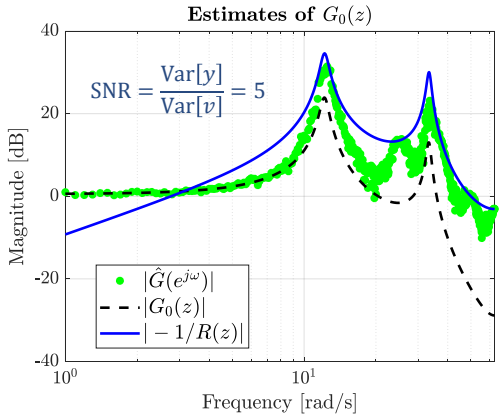


Is there a closed-loop identification problem?



Is there a closed-loop identification problem?

In general, the **nonparametric estimate** $\hat{G}(e^{j\omega})$ is a **compromise** between $G_0(z)$ and $-1/R(z)$



Is there a closed-loop identification problem?

Proof of the nonparametric estimate expression:

The nonparametric estimate is $\hat{G}(e^{j\omega}) = \dfrac{\hat{\Gamma}_{uy}(\omega)}{\hat{\Gamma}_{uu}(\omega)}$

The closed loop equations are

$$\begin{cases} u(t) = S_0(z)[r(t) - R(z)v(t)] \\ y(t) = S_0(z)[G_0(z)r(t) + v(t)] = G_0(z)u(t) + v(t) \end{cases}$$

$$\Gamma_{uu}(\omega) = \left|S_0(e^{j\omega})\right|^2 \left(\Gamma_{rr}(\omega) + \left|R(e^{j\omega})\right|^2 \Gamma_{vv}(\omega)\right)$$

Is there a closed-loop identification problem?

Compute the cross-spectrum $\Gamma_{uy}(\omega)$:

$$u(t)y(t+\tau) = u(t)\big(G_0(z)u(t+\tau) + v(t+\tau)\big) = G_0(z)u(t)u(t+\tau) + u(t)v(t+\tau)$$

$$\Gamma_{uy}(\omega) = G_0(e^{j\omega})\Gamma_{uu}(\omega) + \Gamma_{uv}(\omega)$$

Compute the cross-spectrum $\Gamma_{uv}(\omega)$:

$$u(t+\tau) = r(t+\tau) - R(z)y(t+\tau) = r(t+\tau) - R(z)\big(G_0(z)u(t+\tau) + v(t+\tau)\big)$$

$$= r(t+\tau) - R(z)G_0(z)u(t+\tau) - R(z)v(t+\tau)$$

$$v(t)u(t+\tau) = v(t)\big(r(t+\tau) - R(z)G_0(z)u(t+\tau) - R(z)v(t+\tau)\big)$$

Is there a closed-loop identification problem?

$$v(t)u(t+\tau) = v(t)\big(r(t+\tau) - R(z)G_0(z)u(t+\tau) - R(z)v(t+\tau)\big)$$

$$\Gamma_{vu}(\omega) = -R(e^{j\omega})G_0(e^{j\omega})\Gamma_{vu}(\omega) - R(e^{j\omega})\Gamma_{vv}(\omega) \quad \Rightarrow \quad \Gamma_{vu}(\omega) = -\frac{R(e^{j\omega})}{1 + R(e^{j\omega})G_0(e^{j\omega})}\Gamma_{vv}(\omega)$$

Recall that $\Gamma_{vu}(\omega) = \Gamma_{uv}(-\omega)$ and $\Gamma_{vv}(\omega) = \Gamma_{vv}(-\omega)$, so that

$$\Gamma_{uv}(\omega) = -\frac{R^*(e^{j\omega})}{1 + R^*(e^{j\omega})G_0^*(e^{j\omega})}\Gamma_{vv}(\omega)$$

Is there a closed-loop identification problem?

The expression for $\Gamma_{uy}(\omega)$ becomes

$$\Gamma_{uy}(\omega) = G_0(e^{j\omega})\Gamma_{uu}(\omega) + \Gamma_{uv}(\omega)$$

$$= G_0(e^{j\omega})\left[\left|S_0(e^{j\omega})\right|^2 \left(\Gamma_{rr}(\omega) + \left|R(e^{j\omega})\right|^2 \Gamma_{vv}(\omega)\right)\right] - R^*(e^{j\omega})S^*(e^{j\omega})\Gamma_{vv}(\omega)$$

$$= \left|S_0(e^{j\omega})\right|^2 \left\{G_0(e^{j\omega})\Gamma_{rr}(\omega) + \left[G_0(e^{j\omega})\left|R(e^{j\omega})\right|^2 - R^*(e^{j\omega})S_0^{-1}(e^{j\omega})\right]\Gamma_{vv}(\omega)\right\}$$

$$= \left|S_0(e^{j\omega})\right|^2 \left\{G_0(e^{j\omega})\Gamma_{rr}(\omega) + \left[G_0(e^{j\omega})R(e^{j\omega})R^*(e^{j\omega}) - R^*(e^{j\omega})\left(1 + G_0(e^{j\omega})R(e^{j\omega})\right)\right]\Gamma_{vv}(\omega)\right\}$$

$$\Gamma_{uy}(\omega) = \left|S_0(e^{j\omega})\right|^2 \left\{G_0(e^{j\omega})\Gamma_{rr}(\omega) - R^*(e^{j\omega})\Gamma_{vv}(\omega)\right\}$$

Is there a closed-loop identification problem?

So that

$$\hat{G}(e^{j\omega}) = \frac{\hat{\Gamma}_{uy}(\omega)}{\hat{\Gamma}_{uu}(\omega)} = \frac{|S_0(e^{j\omega})|^2\{G_0(e^{j\omega})\Gamma_{rr}(\omega) - R^*(e^{j\omega})\Gamma_{vv}(\omega)\}}{|S_0(e^{j\omega})|^2(\Gamma_{rr}(\omega) + |R(e^{j\omega})|^2\Gamma_{vv}(\omega))}$$

$$= \frac{G_0(e^{j\omega})\Gamma_{rr}(\omega) - R^*(e^{j\omega})\Gamma_{vv}(\omega)}{\Gamma_{rr}(\omega) + |R(e^{j\omega})|^2\Gamma_{vv}(\omega)}$$

Advanced material

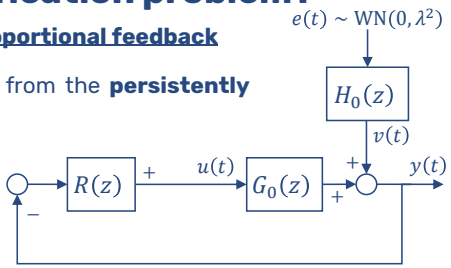
Is there a closed-loop identification problem?

Example: Parametric identification under proportional feedback

Assume $r_1(t) = r_2(t) = 0$. The only excitation is from the **persistently exciting** input $e(t)$

Suppose that the controller is **proportional**, i.e. $R(z) = f$. Then, the control action is

$$u(t) = -f \cdot y(t), \qquad f \in \mathbb{R}$$



Consider a first-order $\mathrm{ARX}(n_a = 1, n_b = 0, k = 1)$ model set so that $\mathcal{S} \in \mathcal{M}$

$$\varepsilon_1(t, \boldsymbol{\theta}) = y(t) - ay(t-1) - bu(t-1), \qquad \boldsymbol{\theta} = [a \quad b]^\mathsf{T}$$

Is there a closed-loop identification problem?

Inserting the feedback law $u(t) = -f \cdot y(t)$ into the model gives

$$\varepsilon_1(t, \boldsymbol{\theta}) = y(t) - ay(t-1) + b(fy(t-1)) = y(t) + (-a + bf)y(t-1)$$

which is the one-step **prediction error** when we apply **data** that are obtained from **closed loop** experiments

From this equation we observe that all models having equal $(-\hat{a} + \hat{b}f)$ give the same prediction error. Thus, we have an **identifiability problem**. If the model set is restricted to

$$\varepsilon_1(t, \boldsymbol{\theta}) = y(t) - ay(t-1) - u(t-1), \qquad \boldsymbol{\theta} = a$$

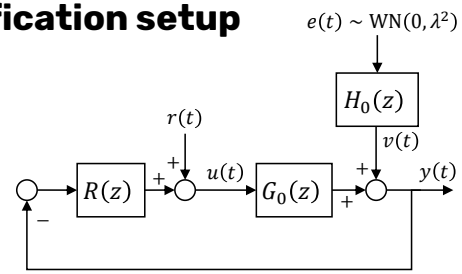
then the data are **sufficiently informative** to uniquely identify the model

A possible solution is to use an **external excitation** $r(t)$

Closed-loop system identification setup

- $\{u(t), y(t)\}$, $t = 1, \dots, N$ measured
- $r_1(t)$ and $r_2(t)$ might be known/measured
- Knowledge of $R(z)$ might be available/used
- $r(t)$ is assumed (quasi) stationary
- $r(t)$ and $v(t)$ are assumed not correlated
- $u(t)$ and $v(t)$ are correlated
- $G_0(z)$ is strictly proper to avoid algebraic loops* (i.e. $y(t)$ depends only on past inputs)

$$r(t) \coloneqq r_1(t) + R(z)r_2(t)$$



Aim: identify $G_0(z)$ and possibly $H_0(z)$

*An algebraic loop occurs if neither $R(z)$ nor $G_0(z)$ contain a delay, i.e. $\lim_{z \to +\infty} R(z)G_0(z) \neq 0$

Closed-loop system identification setup

We will cover the following approaches for closed-loop identification:

1. **Instrumental variables** method
2. **Direct identification** method (based on $u(t)$ and $y(t)$ only)
3. **Indirect identification** methods (based on $u(t), y(t)$ and $r(t)$ and/or $R(z)$)

We will focus on:

- **Consistency** of $G(z, \hat{\boldsymbol{\theta}}_N), H(z, \hat{\boldsymbol{\theta}}_N)$ when $\mathcal{S} \in \mathcal{M}$
- **Consistency** of $G(z, \hat{\boldsymbol{\theta}}_N)$ when $G_0 \in \mathcal{G}$
- Asymptotic estimation **variance**

Outline

Instrumental variables method

The **instrumental variables** approach estimates the open-loop system as an **ARX model** using closed-loop data $u(t), y(t)$ and the signal $r(t)$

An instrumental variable is used to **decorrelate** the regressors $\boldsymbol{\varphi}(t)$ from the noise that enters in the control loop (VRFT Case study)

Thus, the needed information are:

- measurements of $u(t), y(t)$ and $r(t)$

Instrumental variables method

Assume that at least one of the signals $r_1(t)$ or $r_2(t)$ is **measurable**, and denote it as $r(t)$.

Furthermore, assume $r(t)$ **uncorrelated** with the disturbance $e(t)$, so that $r(t) \perp e(t)$

Consider an **ARX model**

$$\mathcal{M}: \; y(t) = \frac{B(z;\boldsymbol{\theta})}{A(z;\boldsymbol{\theta})} u(t-1) + \frac{1}{A(z;\boldsymbol{\theta})} \eta(t) \qquad \eta(t) \sim \mathrm{WN}(0, \lambda^2)$$

$$y(t) = \boldsymbol{\varphi}(t)^\top \boldsymbol{\theta} + \eta(t)$$

$$\boldsymbol{\varphi}(t) = [y(t-1) \;\; \dots \;\; y(t-n_a) \;\; u(t-1) \;\; \dots \;\; u(t-n_b-1)]^\top \in \mathbb{R}^{d \times 1}$$

$$\boldsymbol{\theta} = [a_1 \;\; \dots \;\; a_{n_a} \;\; b_0 \;\; \dots \;\; b_{n_b}]^\top \in \mathbb{R}^{d \times 1}$$

Instrumental variables method

Assume the data are generated by an **ARMAX system**

$$\mathcal{S}:\ y(t) = \frac{B_0(z;\boldsymbol{\theta}^0)}{A_0(z;\boldsymbol{\theta}^0)}u(t-1) + \frac{C_0(z;\boldsymbol{\theta}^0, c_1, \dots, c_{n_c})}{A_0(z;\boldsymbol{\theta}^0)}e(t) \qquad e(t) \sim \mathrm{WN}(0,\lambda^2)$$

$$= G_0(z;\boldsymbol{\theta}^0)u(t-1) + H(z;\boldsymbol{\theta}^0)e(t) \qquad \boldsymbol{\theta}^0 = \begin{bmatrix} a_1^0 & \dots & a_{n_a}^0 & b_0^0 & \dots & b_{n_b}^0 \end{bmatrix}^\top$$

$$y(t) = \boldsymbol{\varphi}(t)^\top \boldsymbol{\theta}^0 + v(t)$$

$$\boldsymbol{\varphi}(t) = [y(t-1) \ \dots \ y(t-n_a) \ u(t-1) \ \dots \ u(t-n_b-1)]^\top \in \mathbb{R}^{d\times 1}$$

$$\boldsymbol{\theta} = \begin{bmatrix} a_1 & \dots & a_{n_a} & b_0 & \dots & b_{n_b} \end{bmatrix}^\top \in \mathbb{R}^{d\times 1}$$

$$v(t) = e(t) + c_1^0 e(t-1) + \cdots + c_{n_c}^0 e(t-n_c)$$

Instrumental variables method

Consider now the **instrumental variable** $\boldsymbol{z}(t) \in \mathbb{R}^{d\times 1}$. The IV estimate is given by

$$\widehat{\boldsymbol{\theta}}_{\mathrm{IV}} = \boldsymbol{\theta}^0 + \left(\sum_{t=1}^{N} \boldsymbol{z}(t)\boldsymbol{\varphi}(t)^\top\right)^{-1} \cdot \sum_{t=1}^{N} \boldsymbol{z}(t)v(t)$$

which is **correct** and **consistent** if:

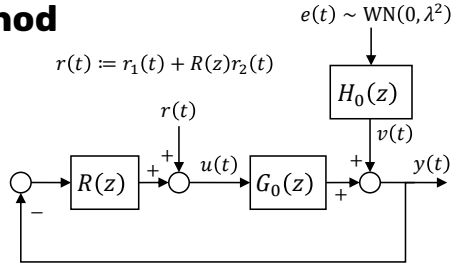
- $\mathbb{E}[\boldsymbol{z}(t)\boldsymbol{\varphi}(t)^\top]$ **is invertible**
- $\mathbb{E}[\boldsymbol{z}(t)v(t)] = \boldsymbol{0}$

Instrumental variables method

In closed-loop, in general, every element of $\boldsymbol{\varphi}(t)$ **is correlated** with $v(t)$, so an instrumental variable is needed

A possible choice is to use an **external signal** $r(t)$ which is **uncorrelated** with $v(t)$, but **correlated** with $\boldsymbol{\varphi}(t)$. Thus

$$\boldsymbol{z}(t) = [r(t-1) \ \ r(t-2) \ \dots \ r(t-d)]^\top \in \mathbb{R}^{d\times 1}$$



Instrumental variables method: conclusions

Within the **IV framework**, we can:

- Estimate **consistently** $G(z,\widehat{\boldsymbol{\theta}}_N), H(z,\widehat{\boldsymbol{\theta}}_N)$ when $\mathcal{S} \in \mathcal{M}$
- Estimate **consistently** $G(z,\widehat{\boldsymbol{\theta}}_N)$ when $G_0 \in \mathcal{G}$
- Estimate **unstable plants** $G_0(z)$

However:

- An **approximation criterion** for the case $G_0 \notin \mathcal{G}$ **can not be given** since the IV method it is not based on an optimization criterion
- The model set is limited to an **ARX structure**

Example: IV method

Consider the $\mathrm{OE}(n_b = 1, n_a = 4, k = 3)$ system sampled at $T_s = 0.05\ s$ with $e(t) \sim \mathrm{WN}(0, \lambda^2)$

$$\mathcal{S}\colon\ y(t) = \frac{(0.28261 + 0.50666z^{-1}) \cdot z^{-3}}{1 - 1.41833z^{-1} + 1.58939z^{-2} - 1.31608z^{-3} + 0.88642z^{-4}} u(t) + e(t)$$

controlled with the regulator

$$R(z) = \frac{0.32905 - 0.59771z^{-1} + 0.70728z^{-2} - 0.64010z^{-3} + 0.46499z^{-4} - 0.11769z^{-5}}{1 - z^{-1}}$$

Let $r_2(t)$ be a full-band PRBS with levels $[-2, 2]$ and $r_1(t) = 0$. Collect $N = 10000$ data in closed-loop with $\mathrm{SNR} = \frac{\mathrm{Var}[y]}{\mathrm{Var}[v]} = 100$

Example: IV method

Identify the open-loop system with an $\mathrm{ARX}(n_a = 4, n_b = 1, k = 3)$ model

$$\mathcal{M}\colon\ y(t) = \frac{(b_0 + b_1z^{-1}) \cdot z^{-3}}{1 + a_1z^{-1} + a_2z^{-2} + a_3z^{-3} + a_4z^{-4}} u(t) + \frac{1}{1 + a_1z^{-1} + a_2z^{-2} + a_3z^{-3} + a_4z^{-4}} e(t)$$

Notice that $\mathcal{S} \notin \mathcal{M}$ but $G_0 \in \mathcal{G}$

- Use a **PEM approach** with $u(t)$ as input and $y(t)$ as output.
- Then, compute an **IV estimate** using $r(t)$ or $r_2(t)$ as instrumental variable. Next plots show results using $r(t) = R(z)r_2(t)$

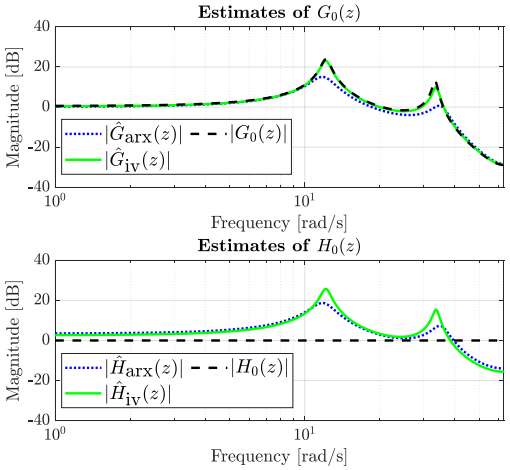
Example: IV method

Estimation of $G_0(z)$:

The **ARX** estimate is **biased** with respect to the IV estimate (in the case $G_0 \in \mathcal{G}$ and $r(t)$ is exciting enough)

Estimation of $H_0(z)$:

Since $\mathcal{S} \notin \mathcal{M}$, the noise model is not estimated correctly (but it is not the main aim of the IV method)



Outline

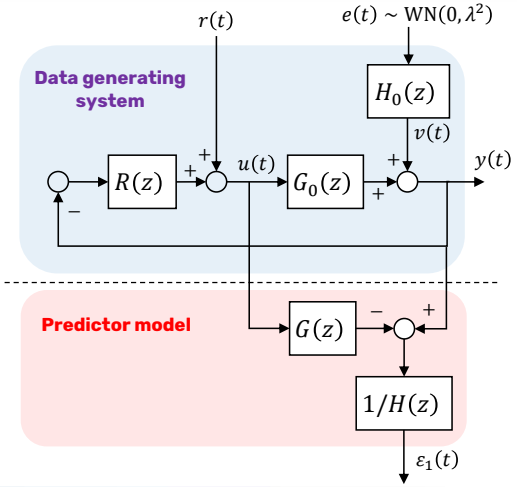
1. Introduction to closed-loop identification
2. Instrumental variables method
3. **Direct identification method**
4. Indirect identification methods

Direct identification

The general idea of **direct identification** is to apply an **open-loop** identification method to data $\{u(t), y(t)\}$ collected in **closed-loop** conditions

Thus, the needed information are:

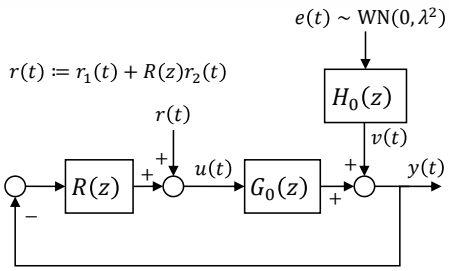
- measurements of $u(t), y(t)$



Direct identification

Consider the closed-loop system equations

$$\begin{cases} u(t) = S_0(z)r(t) - R(z)S_0(z)H_0(z)e(t) \\ y(t) = G_0(z)S_0(z)r(t) + S_0(z)H_0(z)e(t) \end{cases}$$



The **one-step ahead prediction error** for a model $G(z, \boldsymbol{\theta}), H(z, \boldsymbol{\theta})$ is

$$\varepsilon_1(t; \boldsymbol{\theta}) = H^{-1}(z, \boldsymbol{\theta})[y(t) - G(z, \boldsymbol{\theta})u(t)]$$

Direct identification

Substituting the data equations into the prediction error expression, we get

$$\varepsilon_1(t; \boldsymbol{\theta}) = H^{-1}(z, \boldsymbol{\theta})[y(t) - G(z, \boldsymbol{\theta})u(t)]$$

$$= H^{-1}(z, \boldsymbol{\theta})\big[G_0(z)S_0(z)r(t) + S_0(z)H_0(z)e(t) - G(z, \boldsymbol{\theta})\big(S_0(z)r(t) - R(z)S_0(z)H_0(z)e(t)\big)\big]$$

$$= H^{-1}(z, \boldsymbol{\theta})[-G(z, \boldsymbol{\theta})S_0(z)r(t) + G_0(z)S_0(z)r(t) + S_0(z)H_0(z)e(t) + G(z, \boldsymbol{\theta})R(z)S_0(z)H_0(z)e(t)]$$

$$= \frac{S_0(z)}{H(z, \boldsymbol{\theta})}\big(G_0(z) - G(z, \boldsymbol{\theta})\big) \cdot r(t) + \frac{S_0(z)\, H_0(z)}{H(z, \boldsymbol{\theta})}\big(1 + G(z, \boldsymbol{\theta})R(z)\big) \cdot e(t)$$

$$= \frac{S_0(z)}{H(z, \boldsymbol{\theta})}\big(G_0(z) - G(z, \boldsymbol{\theta})\big) \cdot r(t) + \frac{H_0(z)}{H(z, \boldsymbol{\theta})}\frac{S_0(z)}{S(z, \boldsymbol{\theta})} \cdot e(t)$$

Direct identification

$$\varepsilon_1(t; \boldsymbol{\theta}) = \underbrace{\frac{G_0(z) - G(z, \boldsymbol{\theta})}{H(z, \boldsymbol{\theta})\big(1 + R(z)G_0(z)\big)}}_{T_{\varepsilon r}(z, \boldsymbol{\theta})} r(t) + \underbrace{\frac{H_0(z)}{H(z, \boldsymbol{\theta})}\frac{1 + R(z)G(z, \boldsymbol{\theta})}{1 + R(z)G_0(z)}}_{T_{\varepsilon e}(z, \boldsymbol{\theta})} \cdot e(t)$$

Monic if $R(z)G_0(z)$ and $R(z)G(z, \boldsymbol{\theta})$ strictly proper

The **minimum** $\boldsymbol{\theta}^*$ of the **asymptotic cost** $\bar{J} = \mathbb{E}[\varepsilon_1(t)^2]$ is achieved for $N \to +\infty$, providing that $r(t)$ is **persistently exciting** of a sufficient order and $r(t) \perp e(t)$, as

- $T_{\varepsilon r}(z, \boldsymbol{\theta}^*) = 0$
- $T_{\varepsilon e}(z, \boldsymbol{\theta}^*) = 1$

⇨

- $G(z, \boldsymbol{\theta}^*) = G_0(z)$
- $H(z, \boldsymbol{\theta}^*) = H_0(z)$

$\bar{J}(\boldsymbol{\theta}^*) = \mathrm{Var}[e(t)]$

Direct identification when $\mathcal{S} \in \mathcal{M}$

Consistency result for the direct identification method

If $\mathcal{S} \in \mathcal{M}$, the signal $r(t)$ is sufficienty **exciting** and **uncorrelated** with $e(t)$, and there are **no algebraic loops** in the closed-loop system and parametrized models, then

- $G(z, \boldsymbol{\theta}^*) = G_0(z)$
- $H(z, \boldsymbol{\theta}^*) = H_0(z)$

i.e. $G\left(z, \widehat{\boldsymbol{\theta}}_N\right)$ and $H\left(z, \widehat{\boldsymbol{\theta}}_N\right)$ are **consistent estimates**, where $\widehat{\boldsymbol{\theta}}_N$ is the estimate obtained with N data minimizing $\frac{1}{N}\sum_{t=1}^{N} \varepsilon_1(t)^2$

Remark: we **do not need to measure** $r(t)$. It suffices that $r(t)$ is persistently exciting of enough order

Direct identification without external excitation

What happens **when** $r(t)$ **is not present?** Again, consider the one-step prediction error:

$$\varepsilon_1(t;\boldsymbol{\theta}) = \underbrace{\frac{G_0(z) - G(z,\boldsymbol{\theta})}{H(z,\boldsymbol{\theta})\left(1 + R(z)G_0(z)\right)}}_{T_{\varepsilon r}(z,\boldsymbol{\theta})} r(t) + \underbrace{\frac{H_0(z)}{H(z,\boldsymbol{\theta})}\frac{1 + R(z)G(z,\boldsymbol{\theta})}{1 + R(z)G_0(z)}}_{T_{\varepsilon e}(z,\boldsymbol{\theta})} \cdot e(t)$$

Since $r(t)$ is not present, we have to minimize $T_{\varepsilon e}(z, \boldsymbol{\theta})$ so that $T_{\varepsilon e}(z, \boldsymbol{\theta}^*) = 1$. Consistency is achieved if $T_{\varepsilon e}(z, \boldsymbol{\theta}^*) = 1$ **implies** also $T_{\varepsilon r}(z, \boldsymbol{\theta}^*) = 0$

Advanced material

Direct identification without external excitation

The condition $T_{\varepsilon e}(z, \boldsymbol{\theta}^*) = 1$ can be written as

$$\frac{H_0(z)}{H(z,\boldsymbol{\theta})}\frac{1 + R(z)G(z,\boldsymbol{\theta})}{1 + R(z)G_0(z)} = 1 \quad \Rightarrow \quad H^{-1}(z,\boldsymbol{\theta})\left(1 + R(z)G(z,\boldsymbol{\theta})\right) = H_0^{-1}(z)(1 + R(z)G_0(z))$$

Without any additional conditions, this relation is not sufficient to imply $G(z, \boldsymbol{\theta}^*) = G_0(z)$ when $H(z, \boldsymbol{\theta}^*) = H_0(z)$. It depends on the system, the model set, and the controller

However, it can be shown that if the **controller** $R(z)$ has **sufficiently high order** (depending on the data generating system) then **consistent estimates** of $G_0(z)$ and $H_0(z)$ can be obtained

Advanced material

Direct identification without exciting excitation

When $r(t)$ is **present** but **not sufficienty exciting**, data informativity can be guaranteed also by:

- A linear controller of **sufficiently high order**
- A **time-varying** or **nonlinear** controller

Advanced material

Direct identification when $G_0 \in \mathcal{G}$

What happens **when** $\mathcal{S} \notin \mathcal{M}$ **but** $G_0 \in \mathcal{G}$**?** Again, consider the one-step prediction error:

$$\varepsilon_1(t;\boldsymbol{\theta}) = \underbrace{\frac{G_0(z) - G(z,\boldsymbol{\theta})}{H(z,\boldsymbol{\theta})\big(1 + R(z)G_0(z)\big)}}_{T_{\varepsilon r}(z,\boldsymbol{\theta})} r(t) + \underbrace{\frac{H_0(z)}{H(z,\boldsymbol{\theta})}\frac{1 + R(z)G(z,\boldsymbol{\theta})}{1 + R(z)G_0(z)}}_{T_{\varepsilon e}(z,\boldsymbol{\theta})} \cdot e(t)$$

If $H(z,\boldsymbol{\theta}) \neq H_0(z)$, then bringing $T_{\varepsilon e}(z,\boldsymbol{\theta})$ «close to 1» needs to be balanced with making $T_{\varepsilon r}(z,\boldsymbol{\theta})$ «close to 0», since $G(z,\boldsymbol{\theta})$ is present in $T_{\varepsilon e}(z,\boldsymbol{\theta})$. Thus, $\bar{J}(\boldsymbol{\theta}^*) > \mathrm{Var}[e(t)]$

So, if a value $\hat{\boldsymbol{\theta}}$ is such that $T_{\varepsilon r}(z,\hat{\boldsymbol{\theta}}) = 0$, this does not grant that $T_{\varepsilon e}(z,\hat{\boldsymbol{\theta}})$ would be minimized too. So, in general, **no consistency result hold** for the case $G_0 \in \mathcal{G}$

Direct identification of unstable plants

With direct identification using the PEM approach, it is required that the **predictor is stable**. This can be achived even when the model is unstable, but only if it is of type **ARX** or **ARMAX**

In these cases we have (consider the more general **ARMAX** case)

$$\hat{y}(t|t-1;\boldsymbol{\theta}) = H^{-1}(z,\boldsymbol{\theta})G(z,\boldsymbol{\theta})u(t) + \big(1 - H^{-1}(z,\boldsymbol{\theta})\big)y(t)$$

$$= \frac{B(z,\boldsymbol{\theta})}{C(z,\boldsymbol{\theta})}u(t) + \big(1 - A(z,\boldsymbol{\theta})\big)y(t)$$

So that the only condition is on the roots of $C(z,\boldsymbol{\theta})$ and not on those of $A(z,\boldsymbol{\theta})$

Advanced material

Direct identification variance

Assuming $\mathcal{S} \in \mathcal{M}$, the identification **variance** on the **transfer functions** is

$$\mathrm{Var}\big[G(e^{j\omega},\hat{\boldsymbol{\theta}}_N)\big] \approx \frac{n}{N}\cdot\frac{\Gamma_{vv}(\omega)}{\Gamma_{uu}^r(\omega)} = \frac{n}{N}\cdot\frac{\Gamma_{vv}(\omega)}{\Gamma_{uu}(\omega)}\cdot\left[1 + \frac{\Gamma_{uu}^e(\omega)}{\Gamma_{uu}^r(\omega)}\right]$$

$$\mathrm{Var}\big[H(e^{j\omega},\hat{\boldsymbol{\theta}}_N)\big] \approx \frac{n}{N}\cdot\frac{\Gamma_{vv}(\omega)}{\lambda^2}\cdot\frac{\Gamma_{uu}(\omega)}{\Gamma_{uu}^r(\omega)} = \frac{n}{N}\cdot\frac{\Gamma_{vv}(\omega)}{\lambda^2}\cdot\left[1 + \frac{\Gamma_{uu}^e(\omega)}{\Gamma_{uu}^r(\omega)}\right]$$

where n is the order of $\hat{G}(z)$ and $\Gamma_{uu}(\omega) = \Gamma_{uu}^r(\omega) + \Gamma_{uu}^e(\omega)$ with

$$\Gamma_{uu}^r(\omega) = \big|S_0(e^{j\omega})\big|^2 \cdot \Gamma_{rr}(\omega) \qquad \Gamma_{uu}^e(\omega) = \big|R(e^{j\omega})S_0(e^{j\omega})H_0(e^{j\omega})\big|^2 \cdot \lambda^2$$

that denote, respectively, the reference $r(t)$ and noise $e(t)$ contributions to the input $u(t)$

Advanced material

Direct identification variance

Observations:

- In open-loop, $u(t) \perp e(t)$ and so $\Gamma_{uu}^e(\omega) = 0$. Furthermore, $\Gamma_{uu}^r(\omega) = \Gamma_{uu}(\omega)$ and $R(z) = 0$. Then, the open-loop expressions are recovered (IMAD Lesson 12)
- Only the **noise-free part of the input** signal contributes to variance reduction of the estimates
- These expressions do not hold when $\mathcal{S} \notin \mathcal{M}$ with $G_0 \in \mathcal{G}$
- If $G_0 \in \mathcal{G}$ and $H_0(z)$ is **known**, then $\mathrm{Var}\big[G(e^{j\omega},\hat{\boldsymbol{\theta}}_N)\big]$ is the same as in the open-loop case, $\mathrm{Var}\big[G(e^{j\omega},\hat{\boldsymbol{\theta}}_N)\big] \approx \frac{n}{N}\cdot\frac{\Gamma_{vv}(\omega)}{\Gamma_{uu}(\omega)}$

Advanced material

Direct identification: conclusions

With the **direct identification** approach using **PEM**, we get

- **Consistent estimates** if $\mathcal{S} \in \mathcal{M}$ under **excitation** conditions from $r(t)$, or by excitation from the **noise** through a sufficiently **complex controller**
- **No consistency** when only $G_0 \in \mathcal{G}$
- **Periodic excitation** of $u(t)$ is not feasible since not directly controlled and influenced by noise $e(t)$
- **Unstable plants** can be modelled only with particular model structures (ARX, ARMAX)
- In situation of consistency, maximum likelihood results remain valid (Cramer Rao lower bound). But noise models need to be accurately estimated!
- Results remain valid for **nonlinear** and/or **time-varying** controllers

Example: direct identification method

Consider the $\mathrm{BJ}(n_c = 1, n_b = 1, n_d = 1, n_f = 4, k = 3)$ system sampled at $T_s = 0.05\,\mathrm{s}$ with $e(t) \sim \mathrm{WN}(0, \lambda^2)$

$$\mathcal{S}\colon\ y(t) = \frac{B(z)\cdot z^{-3}}{F(z)}u(t) + \frac{C(z)}{D(z)}e(t)$$

$$F(z) = 1 - 1.41833z^{-1} + 1.58939z^{-2} - 1.31608z^{-3} + 0.88642z^{-4}$$

$$B(z) = 0.28261 + 0.50666z^{-1} \qquad C(z) = 1 + 0.134z^{-1} \qquad D(z) = 1 - 0.789z^{-1}$$

controlled with the regulator

$$R(z) = \frac{0.32905 - 0.59771z^{-1} + 0.70728z^{-2} - 0.64010z^{-3} + 0.46499z^{-4} - 0.11769z^{-5}}{1 - z^{-1}}$$

Example: direct identification method

Let $r_2(t)$ be a full-band PRBS with levels $[-2, 2]$ and $r_1(t) = 0$. Collect $N = 10000$ data in closed-loop with $\mathrm{SNR} = \frac{\mathrm{Var}[y]}{\mathrm{Var}[v]} = 100$

Identify $G_0(z)$ and $H_0(z)$ using:

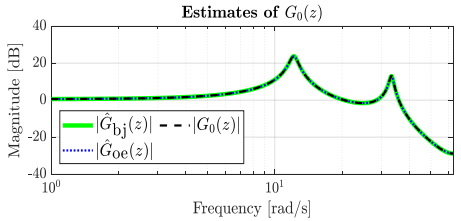
- A **Box-Jenkins** model so that $\mathcal{S} \in \mathcal{M}$
- An **OE** model so that $\mathcal{S} \notin \mathcal{M}$ but $G_0 \in \mathcal{G}$

Example: direct identification method

Try with an **OE model** s.t. $G_0 \in \mathcal{G}$

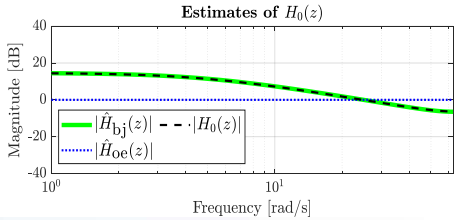
Estimation of $G_0(z)$:

The **OE** estimate is **almost perfect** (apart from high frequencies)



Estimation of $H_0(z)$:

The **OE** estimate does not model the noise term



Example: direct identification method

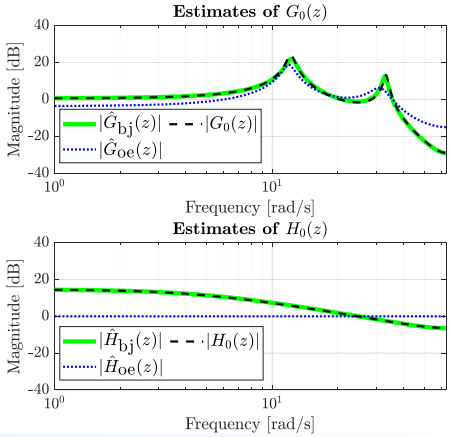
Try with an **OE model** s.t. $G_0 \in \mathcal{G}$ but with $\mathrm{SNR} = 0.1$ instead of $\mathrm{SNR} = 100$

Estimation of $G_0(z)$:

The **OE** estimate is still quite good but **biased**

Estimation of $H_0(z)$:

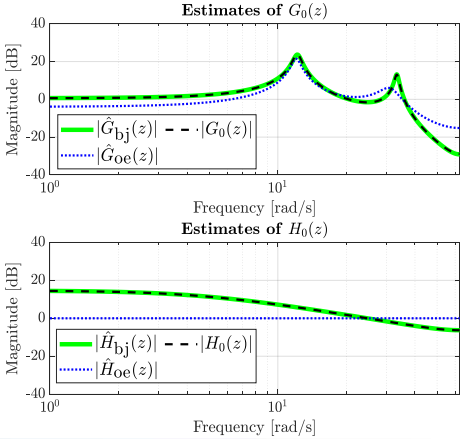
The **OE** estimate does not model the noise term



Example: direct identification method

Let's increase the number of measured data to $N = 100000$

Due to **lack of consistency** since $\mathcal{S} \notin \mathcal{M}$, the **OE model** is **still biased** even if $G_0 \in \mathcal{G}$



Example: direct identification method

Conclusions:

The **bias** (while always present in case $\mathcal{S} \notin \mathcal{M}$) will be **small** if one (or all) of the following holds:

- The noise model is good
- The feedback contribution to the input spectrum $\Gamma_{uu}^{e}(\omega)/\Gamma_{uu}(\omega)$ is small
- The signal to noise ratio is good

In particular, it follows that if a reasonably flexible, **independently parameterized** noise model is used, then the bias-inclination of $\hat{G}(z)$ can be negligible

Outline

Indirect identification

Indirect identification might not use measurements of $u(t)$ inside the closed loop, but it bases its estimate on:

- measurements of the **output** $y(t)$
- measurements of a **«reference»** $r(t)$. We again assume that $r(t)$ is one (or both) of $r_1(t)$ and $R(z)r_2(t)$
- (optionally) knowledge of the **controller** $R(z)$

We will cover the following approaches for indirect closed-loop identification:

1. **General indirect** approach
2. **Comprime factors** approach
3. **Projection/IV** approach

General indirect approach

The **general indirect approach** consists of two steps:

1. **Identify** the **closed-loop system** using $r(t)$ as input and $y(t)$ as output, using a standard (open loop) model set This is a standard open-loop identification problem, as the input signal $r(t)$ is assumed to be uncorrelated with the disturbance signal $e(t)$
2. **Determine** the **open-loop system** parameters from the closed-loop model obtained in step 1, using knowledge of the **controller** $R(z)$

Thus, the needed information are:

- measurements of $y(t)$ and $r(t)$
- knowledge of $R(z)$

General indirect approach

Using the closed-loop equations

$$\begin{cases} u(t) = S_0(z)r(t) - R(z)H_0(z)S_0(z)e(t) \\ y(t) = G_0(z)S_0(z)r(t) + H_0(z)S_0(z)e(t) \end{cases}$$

it follows that the **system** to be identified is a closed-loop one:

$$\mathcal{S}_c: \; y(t) = G_0(z)S_0(z)r(t) + H_0(z)S_0(z)e(t)$$

The **model set** to be used is

$$\mathcal{M}_c: \; y(t) = G_{yr}(z,\boldsymbol{\rho})r(t) + H_{ye}(z,\boldsymbol{\rho})e(t)$$

So we are identifying the transfer functions $G_{yr}^0(z) = G_0(z)S_0(z)$ and $H_{ye}^0(z) = H_0(z)S_0(z)$

General indirect approach

This is basically an open-loop identification problem!

Under the conditions that $\mathcal{S}_c \in \mathcal{M}_c$ and the input **signal** $r(t)$ is **sufficiently exciting**, a PEM approach guarantees **consistency** of the estimates, so that $G_{yr}(z,\boldsymbol{\rho}^*) = G_{yr}^0(z)$ and $H_{ye}(z,\boldsymbol{\rho}^*) = H_{ye}^0(z)$

To find an estimate of $G_0(z)$ and $H_0(z)$ with N data, we need to solve the following equations:

$$\frac{G(z,\hat{\boldsymbol{\theta}})}{1+G(z,\hat{\boldsymbol{\theta}})R(z)} = G_{yr}(z,\hat{\boldsymbol{\rho}}_N) \qquad \frac{H(z,\hat{\boldsymbol{\theta}})}{1+G(z,\hat{\boldsymbol{\theta}})R(z)} = H_{ye}(z,\hat{\boldsymbol{\rho}}_N)$$

General indirect approach

Isolating the unknown terms in the equations we find that an estimate of $G(z,\hat{\boldsymbol{\theta}})$ and $H(z,\hat{\boldsymbol{\theta}})$ as

$$G(z,\hat{\boldsymbol{\theta}}) = \frac{G_{yr}(z,\hat{\boldsymbol{\rho}}_N)}{1 - R(z)G_{yr}(z,\hat{\boldsymbol{\rho}}_N)} \qquad H(z,\hat{\boldsymbol{\theta}}) = \frac{H_{ye}(z,\hat{\boldsymbol{\rho}}_N)}{1 - R(z)G_{yr}(z,\hat{\boldsymbol{\rho}}_N)}$$

If $G_{yr}(z,\hat{\boldsymbol{\rho}}_N)$ and $H_{ye}(z,\hat{\boldsymbol{\rho}}_N)$ are **consistent estimates**, then it can be verified that

$$G(z,\hat{\boldsymbol{\theta}}) = G_0(z) \qquad H(z,\hat{\boldsymbol{\theta}}) = H_0(z)$$

General indirect approach: conclusions

With the **general indirect identification** approach using **PEM**, we get

- **Consistent estimates** of $G(z,\hat{\boldsymbol{\theta}})$ and $H(z,\hat{\boldsymbol{\theta}})$ if $\mathcal{S}_c \in \mathcal{M}_c$ under **excitation** conditions from $r(t)$
- **Consistent estimates** of $G(z,\hat{\boldsymbol{\theta}})$ if $G_{yr} \in \mathcal{G}_c$ and $r(t)$ persistently exciting
- **Tunable** estimation **bias** as for the open-loop identification case
- **Periodic excitations** $r(t)$ are possible
- **Unstable plants** can be modelled

However:

- Knowledge of the **controller** $R(z)$ is required
- The **order** of $G(z,\hat{\boldsymbol{\theta}})$ can not be specified

Coprime factors approach

The **coprime factors approach** consists of three steps:

1. **Identify** the transfer function $G_{yr}^0(z) \coloneqq G_0(z)S_0(z)$ from $r(t)$ to $y(t)$
2. **Identify** the transfer function $G_{ur}^0(z) \coloneqq S_0(z)$ from $r(t)$ to $u(t)$
3. **Compute** $\hat{G}(z) = \hat{G}_{yr}(z)/\hat{G}_{ur}(z)$

Thus, the needed information are:

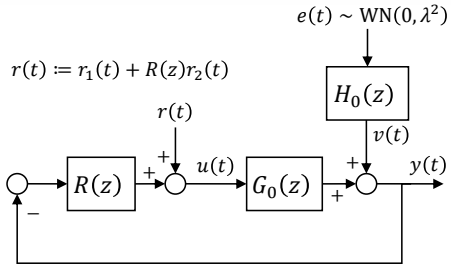
- measurements of $u(t), y(t)$ and $r(t)$

Coprime factors approach

Recall the closed-loop system equations:

$$\begin{cases} u(t) = S_0(z)r(t) - R(z)S_0(z)v(t) \\ y(t) = G_0(z)S_0(z)r(t) + S_0(z)v(t) \end{cases}$$

$$S_0(z) = \frac{1}{1 + R(z)G_0(z)}$$



The transfer functions $r(t) \to \begin{bmatrix} y(t) \\ u(t) \end{bmatrix}$ can be estimated with open-loop methods

PEM estimation of $G_{yr}^0(z)$: $\varepsilon_1^{yr}(t;\boldsymbol{\rho}) = H_{\mathrm{ind}}^{yr}(z,\boldsymbol{\eta})^{-1}\left[y(t) - G_{yr}(z,\boldsymbol{\rho})r(t)\right]$

PEM estimation of $G_{ur}^0(z)$: $\varepsilon_1^{ur}(t;\boldsymbol{\rho}) = H_{\mathrm{ind}}^{ur}(z,\boldsymbol{\eta})^{-1}\left[y(t) - G_{ur}(z,\boldsymbol{\rho})u(t)\right]$

Coprime factors approach

If $G_{yr}^0(z)$ and $G_{ur}^0(z)$ are estimated **consistently**, it follows from

$$G_{yr}^0(z) = G_{yr}(z, \boldsymbol{\rho}_0) = G_0(z)S_0(z)$$

$$G_{ur}^0(z) = G_{ur}(z, \boldsymbol{\rho}_0) = S_0(z)$$

that

$$G(z, \hat{\boldsymbol{\theta}}_N) := \frac{G_{yr}(z, \hat{\boldsymbol{\rho}}_N)}{G_{ur}(z, \hat{\boldsymbol{\rho}}_N)}$$

is a **consistent estimate** of $G_0(z)$. This estimate can also be computed in a nonparametric frequency-domain way using the **Empirical Transfer Function Estimate (ETFE)**

Coprime factors approach

Consistency result for general and coprime factors indirect identification method

If model structures $G_{yr}(z, \boldsymbol{\rho})$ and $G_{ur}(z, \boldsymbol{\rho})$ are chosen such that they **contain the underlying true system** dynamics, and $H_{\text{ind}}^{yr}(z, \boldsymbol{\eta}), H_{\text{ind}}^{ur}(z, \boldsymbol{\eta})$ are **parametrized independently** from $\boldsymbol{\rho}$, then

$\hat{G}(z)$ **is a consistent estimate of** $G_0(z)$

provided that $r(t)$ is **persistently exciting** of a sufficiently high order

Note: we **do not need** to estimate a **noise model** to obtain a consistent estimate of $G_0(z)$

Coprime factors approach: conclusions

With the **coprime factors** approach using **PEM**, we get

- **Consistent estimates** of $G(z, \hat{\boldsymbol{\theta}})$ and $H(z, \hat{\boldsymbol{\eta}})$ if $\mathcal{S}_c \in \mathcal{M}_c$ under **excitation** conditions from $r(t)$
- **Consistent estimates** of $G(z, \hat{\boldsymbol{\theta}})$ if $G_{yr} \in \mathcal{G}_c$ and $r(t)$ persistently exciting
- **Tunable** estimation **bias** as for the open-loop identification case
- **Periodic excitations** $r(t)$ are possible
- **Unstable plants** can be modelled
- Knowledge of **controller not needed**

However:

- The **order** of $G(z, \hat{\boldsymbol{\theta}})$ can not be specified
- The input $u(t)$ must be measured

Example: coprime factors approach

The **nonparametric** estimate has been already shown in a previous example. Consider now a **parametric PEM** estimate of $\text{BJ}(n_c = 1, n_b = 1, n_d = 1, n_f = 4, k = 3)$ system sampled at $T_s = 0.05$ s with $e(t) \sim \text{WN}(0, \lambda^2)$

$$\mathcal{S}\colon\ y(t) = \frac{B(z) \cdot z^{-3}}{F(z)} u(t) + \frac{C(z)}{D(z)} e(t)$$

$$F(z) = 1 - 1.41833z^{-1} + 1.58939z^{-2} - 1.31608z^{-3} + 0.88642z^{-4}$$

$$B(z) = 0.28261 + 0.50666z^{-1} \qquad C(z) = 1 + 0.134z^{-1} \qquad D(z) = 1 - 0.789z^{-1}$$

controlled with the regulator

$$R(z) = \frac{0.32905 - 0.59771z^{-1} + 0.70728z^{-2} - 0.64010z^{-3} + 0.46499z^{-4} - 0.11769z^{-5}}{1 - z^{-1}}$$

Example: coprime factors approach

Let $r_2(t)$ be a full-band PRBS with levels $[-2, 2]$ and $r_1(t) = 0$. Collect $N = 10000$ data in closed-loop with $\text{SNR} = \frac{\text{Var}[y]}{\text{Var}[v]} = 1$

The **output equation** is: $y(t) = G_0(z)S_0(z)r(t) + S_0(z)v(t)$

The function $G_{yr}^0(z) = G_0(z)S_0(z)$ reads as: $G_{yr}^0(z) = \dfrac{0.2826z^{-3} + 0.2241z^{-4} - 0.5067z^{-5}}{\text{Den}(z)}$

The function $S_0(z)$ reads as: $S_0(z) = \dfrac{1 - 2.418z^{-1} + 3.008z^{-2} - 2.905z^{-3} + 2.203z^{-4} - 0.8864z^{-5}}{\text{Den}(z)}$

$$\text{Den}(z) = 1 - 2.418z^{-1} + 3.008z^{-2} - 2.812z^{-3} + 2.2z^{-4} - 0.9894z^{-5} + 0.1775z^{-6} - 0.1929z^{-7} + 0.2023z^{-8} - 0.0596z^{-9}$$

Example: coprime factors approach

The **input equation is:** $u(t) = S_0(z)r(t) - R(z)S_0(z)v(t)$

The function $R(z)S_0(z)$ reads as: $R(z)S_0(z) = \dfrac{\text{Num(z)}}{\text{Den}(z)}$

where both $\text{Num}(z)$ and $\text{Den}(z)$ are 9 degree polynomials, with

$$\text{Den}(z) = 1 - 2.418z^{-1} + 3.008z^{-2} - 2.812z^{-3} + 2.2z^{-4} - 0.9894z^{-5} + 0.1775z^{-6} - 0.1929z^{-7} + 0.2023z^{-8} - 0.0596z^{-9}$$

Identify $G_0(z)$ using:

- A **Box-Jenkins** model so that $\mathcal{S}_c \in \mathcal{M}_c$ (in both identifications cases)
- An **ARMAX** model so that $\mathcal{S}_c \in \mathcal{M}_c$ (in both identifications cases)

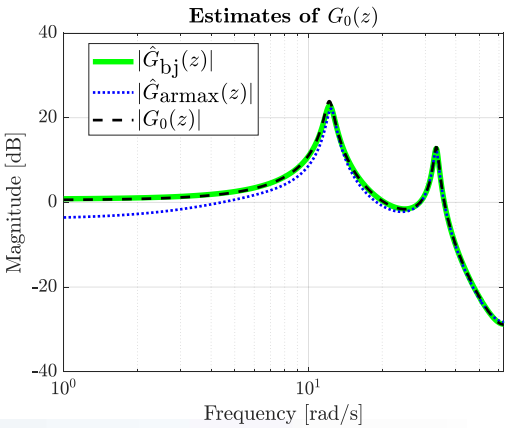
Example: coprime factors approach

Estimation of $G_0(z)$:

The **BJ** estimate is **almost perfect** (but the **order** is **14**!)

The **ARMAX** estimate presents some **bias**

Although in both cases $\mathcal{S}_c \in \mathcal{M}_c$, the BJ model presents the I/O and noise parts **independently parametrized**



Example: coprime factors approach

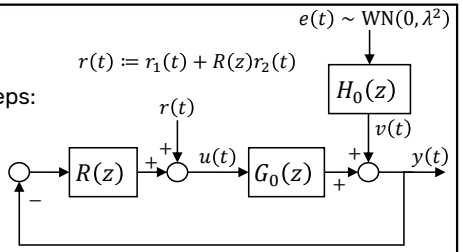
With the coprime factors method, the order of $\hat{G}(z)$ can not be controlled, and usually it turns out to be **very high** after the ratio $\hat{G}(z) = \frac{G_{yr}(z,\hat{\rho}_N)}{G_{ur}(z,\hat{\rho}_N)}$

However, **model reduction** techniques can be used to reduce the complexity of the estimated model (`balred` command in Matlab)

Projection/IV approach

The **projection/IV approach** consists of two steps:

1. **Identify** the transfer function $G_{ur}^0(z) \coloneqq S_0(z)$ from $r(t)$ to $u(t)$



2. **Identify** the transfer function $G_0(z)$ from $u(t)$ to $y(t)$ using the part $u^r(t)$ of $u(t)$ that depends on $r(t)$, i.e. $u^r(t) = S_0(z)r(t)$

Thus, the needed information are:

- measurements of $u(t), y(t)$ and $r(t)$

Projection/IV approach

The main idea is to recognize that the **input** signal $u(t)$ has **two components:**

$$u(t) = u^r(t) + u^e(t)$$

- $u^r(t) = S_0(z)r(t)$: the part of $u(t)$ generated from $r(t)$
- $u^e(t) = -R(z)S_0(z)H_0(z)e(t)$: the part of $u(t)$ generated from $e(t)$

} **Uncorrelated** since $r(t)$ and $e(t)$ are uncorrelated

Then, the data-generating system is

$$\mathcal{S}\colon\ y(t) = G_0(z)u^r(t) + \underbrace{G_0(z)u^e(t) + H_0(z)e(t)}_{\textbf{noise terms}} = G_0(z)u^r(t) + S_0(z)H_0(z)e(t)$$

Idea: Identify $G_0(z)$ using $u^r(t)$ and $y(t)$. This is basically an open-loop problem, although with a greater power of the noise term (i.e. **higher estimation variance**)

Projection/IV approach

Observation

With this approach, the **order** and **structure** of $\hat{G}(z)$ are completely under **our control**

How to get $u^r(t)$?

- **identify** $G_{ur}^0(z) \coloneqq S_0(z)$ using $r(t)$ and $u(t)$, again as an **open-loop** problem as described in previous approaches, obtaining an estimate $G_{ur}(z, \hat{\boldsymbol{\rho}}_N) \coloneqq S(z, \hat{\boldsymbol{\rho}}_N)$
- **simulate**

$$\hat{u}^r(t) = S(z, \hat{\boldsymbol{\rho}}_N)r(t)$$

- use $\hat{u}^r(t)$ as **input signal** and $y(t)$ as output for the identification of $G_0(z)$

Projection/IV approach

Using a model of type

$$\mathcal{M}\colon\ y(t) = G(z, \boldsymbol{\theta})u^r(t) + W(z, \boldsymbol{\eta})e(t)$$

The estimates of $G_0(z)$ and $H_0(z)$ are obtained as:

$$G\left(z, \hat{\boldsymbol{\theta}}_N\right)$$

$$H(z, \hat{\boldsymbol{\eta}}_N) = W(z, \hat{\boldsymbol{\eta}}_N)\hat{S}(z, \hat{\boldsymbol{\rho}}_N)^{-1}$$

Projection/IV approach: conclusions

With the **projection/IV** approach using **PEM**, we get

- **Consistent estimates** of $G(z,\hat{\boldsymbol{\theta}})$ and $H(z,\hat{\boldsymbol{\eta}})$ if $S_0 \in \mathcal{T}$ and $\mathcal{S} \in \mathcal{M}$, under **excitation** conditions from $r(t)$
- **Consistent estimates** of $G(z,\hat{\boldsymbol{\theta}})$ if $S_0 \in \mathcal{T}$, $G_0 \in \mathcal{G}$ and $r(t)$ persistently exciting
- **Tunable** estimation **bias** as for the open-loop identification case
- **Periodic excitations** $r(t)$ are possible
- Knowledge of **controller not needed**
- The **order** and **structure** of $G(z,\hat{\boldsymbol{\theta}})$ can be **imposed**

However:

- **Unstable plants cannot** be modelled
- The input $u(t)$ must be measured

Example: projection/IV approach

Consider a BJ$(n_c = 1, n_b = 1, n_d = 1, n_f = 4, k = 3)$ system sampled at $T_s = 0.05$ s with $e(t) \sim \mathrm{WN}(0, \lambda^2)$

$$\mathcal{S}: y(t) = \frac{B(z)\cdot z^{-3}}{F(z)}u(t) + \frac{C(z)}{D(z)}e(t)$$

$$F(z) = 1 - 1.41833z^{-1} + 1.58939z^{-2} - 1.31608z^{-3} + 0.88642z^{-4}$$

$$B(z) = 0.28261 + 0.50666z^{-1} \qquad C(z) = 1 + 0.134z^{-1} \qquad D(z) = 1 - 0.789z^{-1}$$

controlled with the regulator

$$R(z) = \frac{0.32905 - 0.59771z^{-1} + 0.70728z^{-2} - 0.64010z^{-3} + 0.46499z^{-4} - 0.11769z^{-5}}{1 - z^{-1}}$$

Example: projection/IV approach

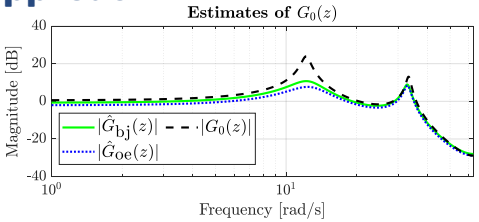
Let $r_2(t)$ be a full-band PRBS with levels $[-2, 2]$ and $r_1(t) = 0$. Collect $N = 10000$ data in closed-loop with $\mathrm{SNR} = \frac{\mathrm{Var}[y]}{\mathrm{Var}[v]} = 0.1$

- Identify $G_{ur}^0(z) \coloneqq S_0(z)$ consistently using PEM and a BJ model of adequate order so that $S_0 \in \mathcal{T}$
- Identify $G_0(z)$ using PEM and a BJ model of adequate order s.t. $\mathcal{S} \in \mathcal{M}$
- Identify $G_0(z)$ using PEM and an OE model of adequate order s.t. $\mathcal{S} \notin \mathcal{M}$ but $G_0 \in \mathcal{G}$

Example: projection/IV approach

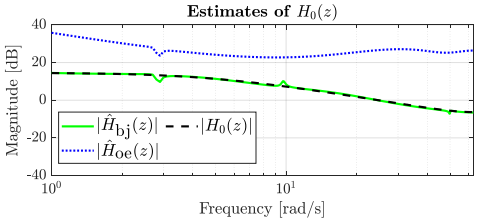
Estimation of $G_0(z)$:

- The **BJ** estimate is (almost) good and similar to the **OE** (compare with direct approach in slide 52)



Estimation of $H_0(z)$:

- The **BJ** estimate is very good
- The **OE** estimate does not model the noise term

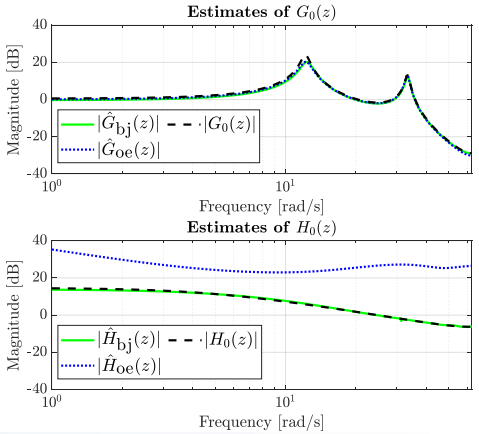


Example: projection/IV approach

Let's increase the number of measured data to $N = 100000$

Due to **consistency** of the indirect approach, both estimates (even when only $G_0 \in \mathcal{G}$) are good (compare with direct method in slide 52)

Furthermore, the **order** of $\hat{G}(z)$ can be controlled as desired



Closed-loop identification summary

	IV	Direct	Indirect		
			General indirect	Coprime factors	Projection/ IV
Consistency of $G(z,\hat{\boldsymbol{\theta}}_N)$ and $H(z,\hat{\boldsymbol{\theta}}_N)$	+	+	+	+	+
Consistency of $G(z,\hat{\boldsymbol{\theta}}_N)$	+	-	+	+	+
Tunable bias (prefiltering)	-	-	+	+	+
Variance	-	+	-	-	--
Fixed model order of $G(z,\hat{\boldsymbol{\theta}}_N)$	+	+	-	-	+
$R(z)$ assumed linear	no	no	yes	yes	yes
$R(z)$ assumed known	no	no	yes	no	no
Measurements of $u(t), y(t)$	yes	yes	$u(t)$ not needed	yes	yes
Measurements of $r(t)$	yes	no	yes	yes	yes

UNIVERSITÀ DEGLI STUDI DI BERGAMO | Dipartimento di Ingegneria Gestionale, dell'Informazione e della Produzione

UNIVERSITÀ DEGLI STUDI DI BERGAMO | Dipartimento di Ingegneria Gestionale, dell'Informazione e della Produzione

ADAPTIVE LEARNING, ESTIMATION AND SUPERVISION OF DYNAMICAL SYSTEMS (ALES)

Lecture 5: Matrix spaces, projections and decompositions

Master Degree in COMPUTER ENGINEERING

Data Science and Data Engineering Curriculum

SPEAKER
Prof. Mirko Mazzoleni

PLACE
University of Bergamo

Syllabus

1. **Recursive and adaptive identification**
   - 1.1 Recursive ARX estimation (RLS)
   - 1.2 Least Mean Squares (LMS)
   - 1.3 Instrumental Variables (IV)
2. **Closed-loop identification**
3. **Subspace and MIMO identification**
   - 3.1 Singular Value Decomposition
   - 3.2 Impulse data: Ho-Kalman, Kung algorithms
   - 3.3 Generic I/O data: the MOESP algorithm
4. **Supervision of dynamical systems**
   - 4.1 Introduction to fault diagnosis
   - 4.2 Model-based fault diagnosis
   - 4.3 Parity space approaches
   - 4.4 Observer-based approaches
   - 4.5 Signal-based fault diagnosis
   - 4.6 Knowledge-based fault diagnosis

CASE STUDIES

- Virtual Reference Feedback Tuning
- Nuclear particles classification
- Leak detection in an industrial valve
- Bearing fault identification

Outline

1. The four fundamental subspaces of a matrix
2. Projections
3. Singular Value Decomposition (SVD)
4. QR decomposition
5. Rank condition and Cayley-Hamilton theorems

Outline

1. **The four fundamental subspaces of a matrix**
2. Projections
3. Singular Value Decomposition (SVD)
4. QR decomposition
5. Rank condition and Cayley-Hamilton theorems

The four fundamental subspaces of a matrix

Consider a generic matrix $A \in \mathbb{R}^{m \times n}$. We can relate to A **four subspaces**

1) Column space (range) $\mathcal{C}(A)$: space of all linear combinations of the **columns** of A. The column space contains vectors in $\mathbb{R}^m$

A useful way to think the product of a **matrix** $A \in \mathbb{R}^{m \times n}$ times a **vector** $x \in \mathbb{R}^{n \times 1}$ is as **linear combinations of the matrix columns**. Take for instance $A \in \mathbb{R}^{3 \times 2}$, $x \in \mathbb{R}^{2 \times 1}$. Then

$$\underset{3\times 2}{A} \cdot \underset{2\times 1}{x} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = x_1 \begin{bmatrix} 1 \\ 3 \\ 5 \end{bmatrix} + x_2 \begin{bmatrix} 2 \\ 4 \\ 6 \end{bmatrix}$$

The result is a vector in $\mathbb{R}^3$ that belongs to $\mathcal{C}(A)$

In this example, the column space of A is **2-dimensional** subspace of $\mathbb{R}^3$

The four fundamental subspaces of a matrix

2) Nullspace (kernel) $\mathcal{N}(A)$: space of all vectors $\boldsymbol{x} \in \mathbb{R}^{n\times 1}$ so that

$$\underset{m\times n}{A}\cdot\underset{n\times 1}{\boldsymbol{x}}=\underset{m\times 1}{\boldsymbol{0}}$$

The null space contains vectors in $\mathbb{R}^n$

3) Row space $\mathcal{C}(A^\top)$: space of all linear combinations of the **rows** of A. The row space contains vectors in $\mathbb{R}^n$

The row space can be generated by considering all **vector-matrix** products. Take for instance $A \in \mathbb{R}^{3\times 2}$, $\boldsymbol{y} \in \mathbb{R}^{3\times 1}$. Then

$$\underset{1\times 3}{\boldsymbol{y}^\top}\cdot\underset{3\times 2}{A}=\begin{bmatrix}y_1 & y_2 & y_3\end{bmatrix}\begin{bmatrix}1 & 2\\ 3 & 4\\ 5 & 6\end{bmatrix}=y_1[1 \quad 2]+y_2[3 \quad 4]+y_3[5 \quad 6]$$

The result is a vector in $\mathbb{R}^2$ that belongs to $\mathcal{C}(A^\top)$

In this example, the row space of A is **2-dimensional** subspace of $\mathbb{R}^2$ (is actually all $\mathbb{R}^2$)

The four fundamental subspaces of a matrix

4) Left nullspace $\mathcal{N}(A^\top)$: space of all vectors $\boldsymbol{y} \in \mathbb{R}^{m\times 1}$ so that

$$\underset{n\times m}{A^\top}\cdot\underset{m\times 1}{\boldsymbol{y}}=\underset{n\times 1}{\boldsymbol{0}}$$

The left nullspace contains vectors in $\mathbb{R}^m$. It can be also interpreted as the set of $\boldsymbol{y}^\top$ s.t.

$$\underset{1\times m}{\boldsymbol{y}^\top}\cdot\underset{m\times n}{A}=\underset{1\times n}{\boldsymbol{0}^\top}$$

The relation $A^\top\cdot\boldsymbol{y}=\boldsymbol{0}$ means that $\boldsymbol{y}$ is **orthogonal** to the **rows** of $A^\top$. Thus, $\boldsymbol{y}$ is orthogonal to the **columns** of A. Then, it follows that

$$\mathcal{C}(A)\perp\mathcal{N}(A^\top)$$

Both subspaces contain vectors in $\mathbb{R}^m$

The four fundamental subspaces of a matrix

Likewise, the vectors in the nullspace $\mathcal{N}(A)$ are **orthogonal** to the **rows** of A, so that

$$\mathcal{C}(A^\top)\perp\mathcal{N}(A)$$

Both subspaces contain vectors in $\mathbb{R}^n$

In particular, the subspaces $\mathcal{C}(A)\perp\mathcal{N}(A^\top)$ and $\mathcal{C}(A^\top)\perp\mathcal{N}(A)$ are not «only» orthogonal: they are **orthogonal complements**

Definition: the **orthogonal complement** of a subspace V contains **every** vector that is perpendicular to V. This orthogonal subspace is denoted $V^\perp$

It follows that, for instance, the row space $\mathcal{C}(A^\top)$ is the orthogonal complement of the null space $\mathcal{N}(A)$. *Every* $\boldsymbol{x}$ perpendicular to the rows satisfies $A\boldsymbol{x}=\boldsymbol{0}$

The four fundamental subspaces of a matrix

The **dimension** of a subspace is the number of its **basis vectors**. The **rank** r is the **number of basis vectors** for the column and row space

The **fundamental theorem of linear algebra** relates the dimensions of the four fundamental subspaces. Given a matrix $A \in \mathbb{R}^{m\times n}$ with rank r, we have that

	$\mathcal{C}(A)$	$\mathcal{N}(A^\top)$	$\mathcal{C}(A^\top)$	$\mathcal{N}(A)$
	$\perp$		$\perp$	
Dimension	r	$m-r$	r	$n-r$
Vectors belong in	$\mathbb{R}^m$	$\mathbb{R}^m$	$\mathbb{R}^n$	$\mathbb{R}^n$

The four fundamental subspaces of a matrix

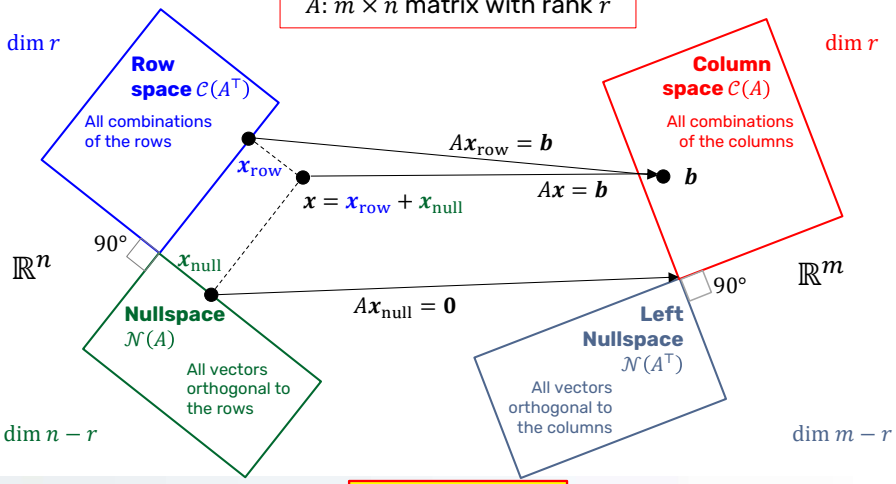
The fact that the four fundamental subspaces are (coupled) orthogonal complements, has important implications. For instance, every $\boldsymbol{x} \in \mathbb{R}^{n\times 1}$ can be split into a **row space** component $\boldsymbol{x}_{\text{row}}$ and a **nullspace** component $\boldsymbol{x}_{\text{null}}$

$$\underset{n\times 1}{\boldsymbol{x}} = \underset{n\times 1}{\boldsymbol{x}_{\text{row}}} + \underset{n\times 1}{\boldsymbol{x}_{\text{null}}}$$

When premultiplied by a matrix $A \in \mathbb{R}^{m\times n}$, we have

$$\underset{m\times 1}{A\boldsymbol{x}} = \underset{m\times 1}{A\boldsymbol{x}_{\text{row}}} + \underset{m\times 1}{A\boldsymbol{x}_{\text{null}}} = \underset{m\times 1}{A\boldsymbol{x}_{\text{row}}}$$

The result belongs to $\mathcal{C}(A)$ by definition. The matrix A can be seen as a linear operator, that, when applied to $\boldsymbol{x}$, projects $\boldsymbol{x}$ into the column space



MATRIX SUSPACES BIG PICTURE

Outline

1. The four fundamental subspaces of a matrix
2. **Projections**
3. Singular Value Decomposition (SVD)
4. QR decomposition
5. Rank condition and Cayley-Hamilton theorems

Orthogonal projection

Assume we have a set of n vectors $\boldsymbol{a}_1, \boldsymbol{a}_2, \dots, \boldsymbol{a}_n \in \mathbb{R}^{m\times 1}$, and assume they are **linearly independent** $(m \geq n)$

We are interested to find the linear combination $\boldsymbol{p} \in \mathbb{R}^{m\times 1}$

$$\boldsymbol{p} = \hat{x}_1\boldsymbol{a}_1 + \hat{x}_2\boldsymbol{a}_2 + \cdots + \hat{x}_n\boldsymbol{a}_n$$

that is **closest** to a given vector $\boldsymbol{b} \in \mathbb{R}^{m\times 1}$. We are (orthogonally) **projecting** $\boldsymbol{b}$ on the subspace spanned by the vectors $\boldsymbol{a}_1, \dots, \boldsymbol{a}_n$

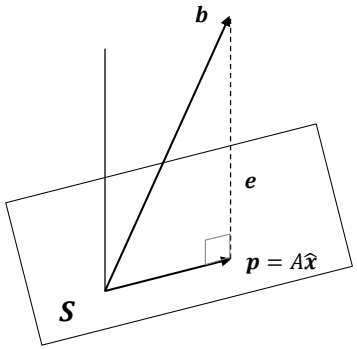
A vector $\boldsymbol{p}$ $(m \times 1)$ can be expressed as $\boldsymbol{p} = A\boldsymbol{x}$, where $A = [\boldsymbol{a}_1 \quad \cdots \quad \boldsymbol{a}_n] \in \mathbb{R}^{m\times n}$. Thus, $\boldsymbol{p}$ belongs to $\mathcal{C}(A)$. We are interested in the particular combination $\boldsymbol{p} = A\hat{\boldsymbol{x}}$ closest to $\boldsymbol{b}$

Orthogonal projection

Let $A \in \mathbb{R}^{3\times 2}$, $\text{rank}(A) = 2$ and $\boldsymbol{S}$ the subspace in $\mathbb{R}^3$ spanned by $\mathcal{C}(A)$

The closest point $\boldsymbol{p}$ on the subspace $\boldsymbol{S}$ is the **orthogonal projection** of $\boldsymbol{b}$ over $\boldsymbol{S}$

The error vector $\boldsymbol{e} = \boldsymbol{b} - \boldsymbol{p} = \boldsymbol{b} - A\hat{\boldsymbol{x}}$ is **perpendicular** to the subspace $\boldsymbol{S}$, i.e. is perpendicular to all basis vectors of $\boldsymbol{S}$ (and their linear combinations)



$$\underset{1\times 3}{\boldsymbol{a}_1^\top}\underset{3\times 1}{(\boldsymbol{b} - A\hat{\boldsymbol{x}})} = 0$$
$$\underset{1\times 3}{\boldsymbol{a}_2^\top}\underset{3\times 1}{(\boldsymbol{b} - A\hat{\boldsymbol{x}})} = 0$$

$$\Rightarrow \quad \underset{2\times 3}{A^\top}\underset{3\times 1}{(\boldsymbol{b} - A\hat{\boldsymbol{x}})} = \underset{2\times 1}{\boldsymbol{0}}$$

Orthogonal projection

In general, we have that the following **orthogonality conditions** should hold

$$\underset{n\times m}{A^\top}\underset{m\times 1}{(\boldsymbol{b} - A\hat{\boldsymbol{x}})} = \underset{n\times 1}{\boldsymbol{0}}$$

Thus, the **combinations** $\hat{\boldsymbol{x}}$ of columns of A that gives the projection $\boldsymbol{p}$ are

$$A^\top(\boldsymbol{b} - A\hat{\boldsymbol{x}}) = \boldsymbol{0} \quad \Rightarrow \quad A^\top\boldsymbol{b} = A^\top A\hat{\boldsymbol{x}} \quad \Rightarrow \quad \hat{\boldsymbol{x}} = (A^\top A)^{-1}A^\top\boldsymbol{b}$$

The **projection** $\boldsymbol{p}$ of $\boldsymbol{b}$ over the subspace spanned by $\mathcal{C}(A)$ is

$$\underset{m\times 1}{\boldsymbol{p}} = \underset{m\times n}{A}\cdot\underset{n\times 1}{\hat{\boldsymbol{x}}} = \underset{m\times m}{A(A^\top A)^{-1}A^\top}\underset{m\times 1}{\boldsymbol{b}} = \underset{m\times m}{P}\cdot\underset{m\times 1}{\boldsymbol{b}}$$

The **projection matrix** is

$$\underset{m\times m}{P} = \underset{m\times n}{A}\underset{n\times n}{(A^\top A)^{-1}}\underset{n\times m}{A^\top}$$

$A^\top A$ is invertible if and only if $\text{rank}(A) = n$

Orthogonal projection

Furthermore, from **orthogonality conditions**

$$\underset{n\times m}{A^\top}\underset{m\times 1}{(\boldsymbol{b} - A\hat{\boldsymbol{x}})} = \underset{n\times 1}{\boldsymbol{0}}$$

we see that $(\boldsymbol{b} - A\hat{\boldsymbol{x}}) = (\boldsymbol{b} - \boldsymbol{p}) = \boldsymbol{e}$ **belongs to the nullspace** of $A^\top$, or equivalently, to the **left nullspace** of A. Remember that $\mathcal{C}(A) \perp \mathcal{N}(A^\top)$

If $P = A(A^\top A)^{-1}A^\top$ is the projection matrix that projects $\boldsymbol{b}$ into $\mathcal{C}(A)$, then the matrix

$$\underset{m\times m}{P^\perp} = \underset{m\times m}{I_m} - \underset{m\times m}{P}$$

projects $\boldsymbol{b}$ into $\mathcal{N}(A^\top)$. In fact

$$P^\perp\boldsymbol{b} = (I_m - P)\boldsymbol{b} = \boldsymbol{b} - \boldsymbol{p} = \boldsymbol{e}$$

so that $P^\perp\boldsymbol{b} = \boldsymbol{e}$ belongs to $\mathcal{N}(A^\top)$

Orthogonal projection

Summarizing, given $A \in \mathbb{R}^{m\times n}$

If $\text{rank}(A) = n$

$$\underset{m\times m}{P_{\text{col}}} := \underset{m\times n}{A}\underset{n\times n}{(A^\top A)^{-1}}\underset{n\times m}{A^\top} = \text{projection matrix onto the } \textbf{column space} \text{ of } A, \text{ i.e. } \mathcal{C}(A)$$

If $\text{rank}(A) = m$

$$\underset{n\times n}{P_{\text{row}}} := \underset{n\times m}{A^\top}\underset{m\times m}{(AA^\top)^{-1}}\underset{m\times n}{A} = \text{projection matrix onto the } \textbf{row space} \text{ of } A, \text{ i.e. } \mathcal{C}(A^\top)$$

- $\boldsymbol{p}_{\text{col}} = P_{\text{col}}\cdot\boldsymbol{b}$ is the orthogonal projection of $\boldsymbol{b} \in \mathbb{R}^m$ onto $\mathcal{C}(A)$
- $\boldsymbol{p}_{\text{row}} = \boldsymbol{c}\cdot P_{\text{row}}$ is the orthogonal projection of $\boldsymbol{c} \in \mathbb{R}^n$ onto $\mathcal{C}(A^\top)$

Orthogonal projection

Summarizing, given $A \in \mathbb{R}^{m \times n}$

If $\text{rank}(A) = n$

$$\underset{m \times m}{P_{\text{col}}^{\perp}} = I_m - \underset{m \times m}{P_{\text{col}}} = \text{projection matrix onto the } \textbf{left nullspace} \text{ of } A, \text{ i.e. } \mathcal{N}(A^{\mathsf{T}})$$

If $\text{rank}(A) = m$

$$\underset{n \times n}{P_{\text{row}}^{\perp}} \coloneqq I_n - \underset{n \times n}{P_{\text{row}}} = \text{projection matrix onto the } \textbf{nullspace} \text{ of } A, \text{ i.e. } \mathcal{N}(A)$$

- $\boldsymbol{p}_{\text{col}}^{\perp} = P_{\text{col}}^{\perp} \cdot \boldsymbol{b} \in \mathcal{N}(A^{\mathsf{T}})$ is orthogonal to $\boldsymbol{p}_{\text{col}} = P_{\text{col}} \cdot \boldsymbol{b} \in \mathcal{C}(A)$, with $\boldsymbol{b} \in \mathbb{R}^m$
- $\boldsymbol{p}_{\text{row}}^{\perp} = \boldsymbol{c} \cdot P_{\text{row}}^{\perp} \in \mathcal{N}(A)$ is orthogonal to $\boldsymbol{p}_{\text{row}} = \boldsymbol{c} \cdot P_{\text{row}} \in \mathcal{C}(A^{\mathsf{T}})$

Example of projection

Consider the matrix $A \in \mathbb{R}^{3 \times 2}$ with $\text{rank}(A) = 2$. Then, $\mathcal{C}(A) \subset \mathbb{R}^3$

$$A = \begin{bmatrix} 0.8147 & 0.9134 \\ 0.9058 & 0.6324 \\ 0.1270 & 0.0975 \end{bmatrix}$$

Compute the projection matrix P_{col} onto $\mathcal{C}(A)$

$$P_{\text{col}} \coloneqq A(A^{\mathsf{T}}A)^{-1}A^{\mathsf{T}} = \begin{bmatrix} 0.993 & -0.003 & 0.0254 \\ -0.003 & 0.9865 & 0.1153 \\ 0.0254 & 0.1153 & 0.0142 \end{bmatrix}$$

Given $\boldsymbol{b} = [0.2785 \quad 0.5469 \quad 0.9575]^{\mathsf{T}} \in \mathbb{R}^3$, its orthogonal projection onto $\mathcal{C}(A)$ is $\boldsymbol{p} = P_{\text{col}}\boldsymbol{b} = [0.3010 \quad 0.6491 \quad 0.0837]^{\mathsf{T}}$ which is the closest vector to $\boldsymbol{b}$ that lies in $\mathcal{C}(A)$

Example of projection

You can verify that $\boldsymbol{p}$ is the vector in $\mathcal{C}(A)$ that is closest to $\boldsymbol{b} \in \mathbb{R}^3$ also computationally, by minimizing the norm between $\boldsymbol{p} = A\hat{\boldsymbol{x}}$ and $\boldsymbol{b}$, solving for $\hat{\boldsymbol{x}}$

In MatLab:

Compute the point in $\mathcal{C}(A) \subseteq \mathbb{R}^2$ closest to $\boldsymbol{b} \in \mathbb{R}^3$

```
min_dist = @(x) (norm(A*x-b)); % define cost function to be minimized
x0 = rand(2,1); % initialize x
xhat = fminsearch(min_dist, x0);
phat = A*xhat % this is equal to p = A*inv(A'*A)*A'*b
```


Example of projection: least squares predictions

Assume $A \in \mathbb{R}^{m \times n}$ is **full rank** $r = n$. In **least squares**, we want to minimize the quantity

$$\underset{m \times n}{\|A} \cdot \underset{n \times 1}{\boldsymbol{x}} - \underset{m \times 1}{\boldsymbol{b}}\|_2^2 = \|\boldsymbol{e}\|_2^2$$

i.e. we want to find $\hat{\boldsymbol{x}}$ so that $\boldsymbol{p} = A\hat{\boldsymbol{x}}$ is **closest** (in squared lenght) to $\boldsymbol{b}$. Again, $\boldsymbol{p}$ can be thought as the projection of $\boldsymbol{b}$ on $\mathcal{C}(A)$

We already know from least squares (and now from projections) that the solution $\hat{\boldsymbol{x}}$ is

$$\hat{\boldsymbol{x}} = (A^{\mathsf{T}}A)^{-1}A^{\mathsf{T}}\boldsymbol{b}$$

So that **the estimated linear model output** is exactly $\boldsymbol{p}$

$$\underset{m \times 1}{\hat{Y}} \coloneqq \boldsymbol{p} = A\hat{\boldsymbol{x}} = A(A^{\mathsf{T}}A)^{-1}A^{\mathsf{T}}\boldsymbol{b} = P \cdot \boldsymbol{b}$$

Example of projection: least squares predictions

The residual error

$$\boldsymbol{e} = A\hat{\boldsymbol{x}} - \boldsymbol{b} = \boldsymbol{p} - \boldsymbol{b}$$

belongs to $\mathcal{N}(A^\top)$ and is orthogonal to $\mathcal{C}(A)$. This can be seen also by taking the gradient of the least squares cost function:

$$J(\boldsymbol{x}) = \|A \cdot \boldsymbol{x} - \boldsymbol{b}\|_2^2 = (A\boldsymbol{x} - \boldsymbol{b})^\top (A\boldsymbol{x} - \boldsymbol{b})$$

Compute the gradient and equal it to zero to find the stationary points (minimum) of $J(\boldsymbol{x})$

$\nabla J(\boldsymbol{x}) = 2A^\top \cdot (A\boldsymbol{x} - \boldsymbol{b}) = 0$ ⇨ The solution $\hat{\boldsymbol{x}}$ is such that $\boldsymbol{e} = A\hat{\boldsymbol{x}} - \boldsymbol{b}$ is orthogonal to $A^\top$, i.e. $\boldsymbol{e} \in \mathcal{N}(A^\top)$

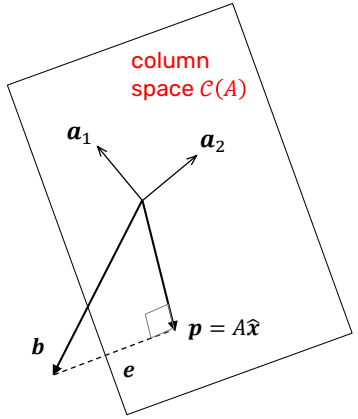
Example of projection: least squares predictions

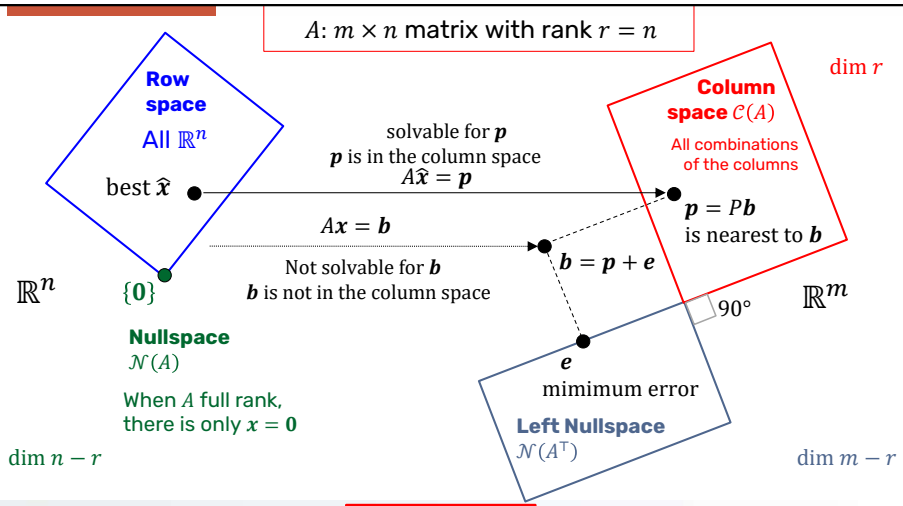
The **estimated linear model output**

$$\underset{m\times 1}{\hat{Y}} \coloneqq \boldsymbol{p} = A\hat{\boldsymbol{x}} = A(A^\top A)^{-1}A^\top \boldsymbol{b} = P \cdot \boldsymbol{b}$$

Is the projection of measured output data $Y \coloneqq \boldsymbol{b}$ on $\mathcal{C}(A)$

When $A\boldsymbol{x} = \boldsymbol{b}$ **is not solvable, i.e. when** $\boldsymbol{b} \notin \mathcal{C}(A)$, then $\boldsymbol{p} = P\boldsymbol{b}$ is the closest approximation to $\boldsymbol{b}$, and $\hat{\boldsymbol{x}}$ is the solution s.t. $\|A \cdot \hat{\boldsymbol{x}} - \boldsymbol{b}\|_2^2$ is minimum





LEAST SQUARES BIG PICTURE

Orthogonal projection eigenvalues

Given a matrix $A \in \mathbb{R}^{m\times n}$ with $m = n$ (i.e. **square matrix**) an eigenvector $\boldsymbol{q} \in \mathbb{R}^{n\times 1}$ is a special vector that **does not change direction** when multiplied by A, i.e.

$$A\boldsymbol{q} = \lambda\boldsymbol{q}$$

where $\lambda \in \mathbb{R}$ is the corresponding **eigenvalue**. When $\lambda = 0$, then $A\boldsymbol{q} = 0\boldsymbol{q}$, so $\boldsymbol{q} \in \mathcal{N}(A)$

A projection matrix satisfies $P = P^2$. If a point is already projected, projecting the point again does not change it. Thus, for a projection matrix, it holds that

$P\boldsymbol{q} = \lambda\boldsymbol{q}$ ⇨ $P \cdot P\boldsymbol{q} = P \cdot \lambda\boldsymbol{q}$ ⇨ $P \cdot P\boldsymbol{q} = \lambda(P\boldsymbol{q})$ ⇨ $P\boldsymbol{q} = \lambda(\lambda\boldsymbol{q})$ ⇨ $\lambda\boldsymbol{q} = \lambda^2\boldsymbol{q}$

So that the **solutions (eigenvalues) of** P **are only** $\lambda = 0$ or $\lambda = 1$

Orthogonal projection eigenvalues

It follows that:

- Vectors $\boldsymbol{x} \in \mathbb{R}^{n\times 1}$ that are in the **direction of the eigenvectors for which** $\lambda = 1$ (or that lie in their span), will **not be modified** by the projection (since $\lambda = 1$)
- Vectors $\boldsymbol{x} \in \mathbb{R}^{n\times 1}$ that are in the **direction of the eigenvectors for which** $\lambda = 0$ (or that lie in their span), will be **projected to zero** (since $\lambda = 0$)

This will be useful when talking about the pseudoinverse matrix

Outline

Singular Value Decomposition

The **Singular Value Decomposition** (SVD) of a matrix $A \in \mathbb{R}^{m\times n}$ of **rank** r is defined as

$$\underset{m\times n}{A} = \underset{m\times m}{U} \cdot \underset{m\times n}{\Sigma} \cdot \underset{n\times n}{V^{\top}}$$

The SVD separates any matrix $A \in \mathbb{R}^{m\times n}$ of rank r into a sum of **rank-1 matrices**

$$A = \begin{bmatrix} | & | & | & | & | \\ \mathbf{u}_1 & \cdots & \mathbf{u}_r & \cdots & \mathbf{u}_m \\ | & | & | & | & | \end{bmatrix} \cdot \begin{bmatrix} \sigma_1 & & & \\ & \ddots & & \\ & & \sigma_r & \\ & & & \end{bmatrix} \cdot \begin{bmatrix} — & \mathbf{v}_1^{\top} & — \\ & \vdots & \\ — & \mathbf{v}_r^{\top} & — \\ & \vdots & \\ — & \mathbf{v}_n^{\top} & — \end{bmatrix}$$

Most «important» part in reconstructing A

$$= \underbrace{\sigma_1 \mathbf{u}_1 \mathbf{v}_1^{\top}}_{\text{rank } 1} + \underbrace{\sigma_2 \mathbf{u}_2 \mathbf{v}_2^{\top}}_{\text{rank } 1} + \cdots + \underbrace{\sigma_r \mathbf{u}_r \mathbf{v}_r^{\top}}_{\text{rank } 1} = \text{rank } r$$

Singular Value Decomposition

- The columns $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_r$ of U form an **orthonormal** basis of $\mathcal{C}(A)$ **(column space** of A in $\mathbb{R}^m$**)**. They are the **eigenvectors** of the symmetric matrix $AA^{\top} \in \mathbb{R}^{m\times m}$
- The columns $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_r$ of V form an **orthonormal** basis of $\mathcal{C}(A^{\top})$ **(row space** of A in $\mathbb{R}^n$**)**. They are the **eigenvectors** of the symmetric matrix $A^{\top}A \in \mathbb{R}^{n\times n}$
- The diagonal elements $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r$ in Σ are > 0 and are called **singular values** of A. They are the **square root of the eigenvalues** of $\frac{1}{N}AA^{\top} \in \mathbb{R}^{m\times m}$ and $\frac{1}{N}A^{\top}A \in \mathbb{R}^{n\times n}$

$$\underset{m\times m}{U} = \begin{bmatrix} | & | & | & | & | \\ \mathbf{u}_1 & \cdots & \mathbf{u}_r & \cdots & \mathbf{u}_m \\ | & | & | & | & | \end{bmatrix}$$

$$\underset{n\times n}{V^{\top}} = \begin{bmatrix} — & \mathbf{v}_1^{\top} & — \\ & \vdots & \\ — & \mathbf{v}_r^{\top} & — \\ & \vdots & \\ — & \mathbf{v}_n^{\top} & — \end{bmatrix}$$

$$\underset{m\times n}{\Sigma} = \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_r \\ & & \end{bmatrix}$$

Singular Value Decomposition

- The columns $\mathbf{u}_{r+1}, \dots, \mathbf{u}_m$ of U form an **orthonormal** basis of $\mathcal{N}(A^\top)$ **(left nullspace space** of A in $\mathbb{R}^m$**)**. They are orthogonal to $\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_r$

  This means that $A^\top \mathbf{u}_i = \mathbf{0}$, for $i > r$

$$U_{m \times m} = \begin{bmatrix} | & | & | & | & | \\ \mathbf{u}_1 & \cdots & \mathbf{u}_r & \cdots & \mathbf{u}_m \\ | & | & | & | & | \end{bmatrix}$$

- The columns $\mathbf{v}_{r+1}, \dots, \mathbf{v}_n$ of V form an **orthonormal** basis of $\mathcal{N}(A)$ **(nullspace** of A in $\mathbb{R}^n$**)**. They are orthogonal to $\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_r$

  This means that $A\mathbf{v}_i = \mathbf{0}$, for $i > r$

$$V^\top_{n \times n} = \begin{bmatrix} — & \mathbf{v}_1^\top & — \\ & \vdots & \\ — & \mathbf{v}_r^\top & — \\ & \vdots & \\ — & \mathbf{v}_n^\top & — \end{bmatrix}$$

Singular Value Decomposition

$$A = \begin{bmatrix} | & | & | & | & | \\ \mathbf{u}_1 & \cdots & \mathbf{u}_r & \cdots & \mathbf{u}_m \\ | & | & | & | & | \end{bmatrix} \cdot \begin{bmatrix} \sigma_1 & & & & \\ & \ddots & & & \\ & & \sigma_r & & \end{bmatrix} \cdot \begin{bmatrix} — & \mathbf{v}_1^\top & — \\ & \vdots & \\ — & \mathbf{v}_r^\top & — \\ & \vdots & \\ — & \mathbf{v}_n^\top & — \end{bmatrix} = \begin{bmatrix} U_r & \bar{U}_r \end{bmatrix} \cdot \Sigma \cdot \begin{bmatrix} V_r^\top \\ \bar{V}_r^\top \end{bmatrix}$$

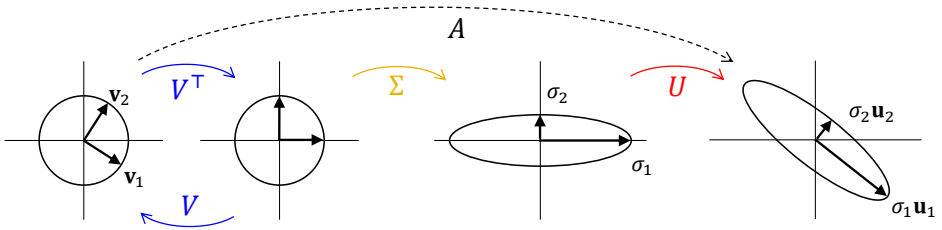
Summarizing:

$$\begin{aligned} \text{Span}(A) &= \text{Span}(U_r) \\ \text{Null}(A^\top) &= \text{Span}(\bar{U}_r) \\ \text{Span}(A^\top) &= \text{Span}(V_r) \\ \text{Null}(A) &= \text{Span}(\bar{V}_r) \end{aligned}$$

Singular Value Decomposition: geometrical view

The SVD **«decomposes» the linear transformation** that a matrix $A \in \mathbb{R}^{m \times n}$ operates on a vector $\boldsymbol{x} \in \mathbb{R}^{n \times 1}$, i.e. $A\boldsymbol{x} = U\Sigma V^\top \boldsymbol{x}$. This can be seen **geometrically**

The SVD separates a matrix into three steps: (**orthogonal**) × (**diagonal**) × (**orthogonal**)
This is equal to say: (**rotation**) × (**stretching**) × (**rotation**)



Matrix norm and rank-k approximation

The highest singuar value σ_1 is the **largest growth factor** of any vector $\boldsymbol{x}$. This is also the **«norm»** of a matrix A *(there are many definitions of matrix norm)*

$$\|A\| = \max_{\boldsymbol{x} \neq \mathbf{0}} \frac{\|A\boldsymbol{x}\|}{\|\boldsymbol{x}\|} = \sigma_1$$

Using the previous norm definition, the **«closest»** rank k matrix to A is

$$A_k = \sigma_1 \mathbf{u}_1 \mathbf{v}_1^\top + \sigma_2 \mathbf{u}_2 \mathbf{v}_2^\top + \cdots + \sigma_k \mathbf{u}_k \mathbf{v}_k^\top$$

that is

$$\|A - B\| \geq \|A - A_k\| = \sigma_{k+1} \text{ for all matrices } B \text{ of rank } k$$

Pseudoinverse

The decomposition $A = U\Sigma V^\top$ can be written as $AV = U\Sigma$. Considering single vectors $\mathbf{v}_i, \mathbf{u}_i$, we have that

$$\underset{m\times n}{A}\cdot\underset{n\times 1}{\mathbf{v}_i} = \sigma_i \cdot \underset{m\times 1}{\mathbf{u}_i}$$

A multiplies $\mathbf{v}_i$ in the $\mathcal{C}(A^\top)$ **(row space of A)** to give $\sigma_i\mathbf{u}_i$ in $\mathcal{C}(A)$ **(column space** of A**)**

Assume $m = n$. **If it exists**, the matrix A^{-1} does the opposite: $A^{-1}\mathbf{u}_i = \mathbf{v}_i/\sigma_i$. The bases are the opposite: the $\mathbf{u}_i$ are in the row space of A^{-1} and the $\mathbf{v}_i$ are in the column space of A^{-1}. We could write (if A^{-1} exists and since U, V are orthogonal)

$$\underset{n\times n}{A^{-1}} = \underset{n\times n}{V}\underset{n\times n}{\Sigma^{-1}}\underset{n\times n}{U^\top}$$

Pseudoinverse

Let $A \in \mathbb{R}^{m\times n}$ a matrix with rank r. Then, A^{-1} does not exist. However, a matrix that multiplies $\mathbf{u}_i$ and gives $\mathbf{v}_i/\sigma_i$ still exists and it is the **pseudoinverse** $A^\dagger \in \mathbb{R}^{n\times m}$ with rank r

$$\underset{n\times m}{A^\dagger} = \underset{n\times n}{V}\cdot\underset{n\times m}{\Sigma^\dagger}\cdot\underset{m\times m}{U^\top}$$

$$A^\dagger = \begin{bmatrix} | & | & | & | & | \\ \mathbf{v}_1 & \cdots & \mathbf{v}_r & \cdots & \mathbf{v}_m \\ | & | & | & | & | \end{bmatrix} \cdot \begin{bmatrix} \sigma_1^{-1} & & & & \\ & \ddots & & & \\ & & \sigma_r^{-1} & & \end{bmatrix} \cdot \begin{bmatrix} — & \mathbf{u}_1^\top & — \\ & \vdots & \\ — & \mathbf{u}_r^\top & — \\ & \vdots & \\ — & \mathbf{u}_n^\top & — \end{bmatrix}$$

Pseudoinverse

Then:

$$\underset{n\times m}{A^\dagger}\underset{m\times 1}{\mathbf{u}_i} = \frac{1}{\sigma_i}\underset{n\times 1}{\mathbf{v}_i}, \quad \text{for } i \le r \qquad \underset{n\times m}{A^\dagger}\underset{m\times 1}{\mathbf{u}_i} = \underset{n\times 1}{\mathbf{0}}, \quad \text{for } i > r$$

- Vectors $\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_r$ in $\mathcal{C}(A)$ **go back** to $\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_r$ in $\mathcal{C}(A^\top)$
- Vectors $\mathbf{u}_{r+1}, \dots, \mathbf{u}_m$ in $\mathcal{N}(A^\top)$ **go to zero**

Pseudoinverse

The product $\Sigma^\dagger\Sigma$ is **as close the identity as we can get** $\underset{n\times m}{\Sigma^\dagger}\underset{m\times n}{\Sigma} = \begin{bmatrix} I_r & \\ & \mathbf{0}_{r-n} \end{bmatrix}$

It follows that

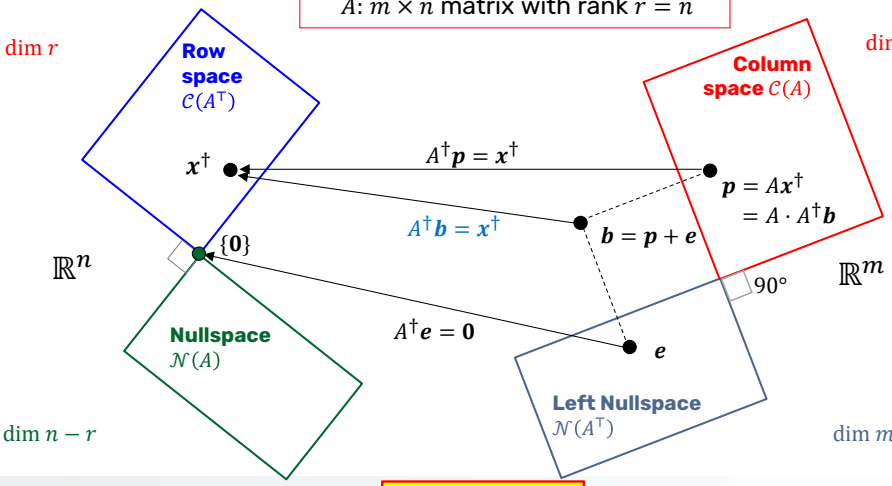
$$A \cdot A^\dagger = U\Sigma V^\top \cdot V\Sigma^\dagger U^\top = U\Sigma\cdot\Sigma^\dagger U^\top$$

so that $A \cdot A^\dagger$ is a **projection matrix**, since it has, as eigenvalues, only $\lambda = 1$ or $\lambda = 0$

$\underset{m\times m}{P_{\text{col}}} \coloneqq \underset{m\times n}{A}\cdot\underset{n\times m}{A^\dagger}$ = projection matrix onto the **column space** of A, i. e. $\mathcal{C}(A)$

$\underset{n\times n}{P_{\text{row}}} \coloneqq \underset{n\times m}{A^\dagger}\cdot\underset{m\times n}{A}$ = projection matrix onto the **row space** of A, i. e. $\mathcal{C}(A^\top)$

For a matrix $A \in \mathbb{R}^{m\times n}$ with **linearly independent columns**, $A^\dagger = (A^\top A)^{-1}A^\top$



PSEUDOINVERSE BIG PICTURE

Example of projection

Consider the matrix $A \in \mathbb{R}^{3\times2}$ with $\text{rank}(A) = 2$. Then, $\mathcal{C}(A) \subset \mathbb{R}^3$ and $\mathcal{C}(A^\mathsf{T}) = \mathbb{R}^2$

$$A = \begin{bmatrix} 0.8147 & 0.9134 \\ 0.9058 & 0.6324 \\ 0.1270 & 0.0975 \end{bmatrix}$$

Compute the projection matrix P_{row} onto $\mathcal{C}(A^\mathsf{T})$. We cannot use the formula $P_{\text{row}} \coloneqq A^\mathsf{T}(AA^\mathsf{T})^{-1}A$ because AA^T is not invertible. Thus, we use the pseudoinverse

$$P_{\text{row}} \coloneqq A^\dagger \cdot A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

The result is an identity matrix! This happens because the rows of A already span all $\mathbb{R}^2$, so there is nothing to project onto. All points in $\mathbb{R}^2$ are already in $\mathcal{C}(A^\mathsf{T})$

Example of projection

The projection matrix P_{col} onto $\mathcal{C}(A)$, computed using the pseudoinverse, is the same as in the previous example, since $\text{rank}(A) = n$

$$P_{\text{col}} \coloneqq A \cdot A^\dagger = A(A^\mathsf{T}A)^{-1}A^\mathsf{T} = \begin{bmatrix} 0.993 & -0.003 & 0.0254 \\ -0.003 & 0.9865 & 0.1153 \\ 0.0254 & 0.1153 & 0.0142 \end{bmatrix}$$

Example of pseudoinverse: least squares estimate

When $A \in \mathbb{R}^{m\times n}, m > n$, is **not full column rank**, $A^\mathsf{T}A$ is **not invertible**. The minimizer of

$$\underset{m\times n}{\|A} \cdot \underset{n\times 1}{\boldsymbol{x}} - \underset{m\times 1}{\boldsymbol{b}}\|_2^2 = \|\boldsymbol{e}\|_2^2$$

can be computed by the **pseudoinverse** $A^\dagger \in \mathbb{R}^{n\times m}$, that will choose, amongst infinitely many possible solutions, that one $x^\dagger$ with **lowest norm**

$$\underset{n\times 1}{\boldsymbol{x}^\dagger} = \underset{n\times m}{A^\dagger} \cdot \underset{m\times 1}{\boldsymbol{b}}$$

The pseudoinverse takes $\mathcal{C}(A)$ **back to** $\mathcal{C}(A^\mathsf{T})$ - both have dimension r. The **error** e is **killed in the nullspace** $\mathcal{N}(A)$

Outline

1. The four fundamental subspaces of a matrix
2. Projections
3. Singular Value Decomposition (SVD)
4. **QR decomposition**
5. Rank condition and Cayley-Hamilton theorems

QR decomposition

Any matrix $A \in \mathbb{R}^{m \times n}$ can be decomposed as

$$\underset{m \times n}{A} = \underset{m \times m}{Q} \cdot \underset{m \times n}{R}$$

- Q is an **orthogonal** matrix, $Q^\top Q = I_m$
- R is **upper triangular** 

  augmented with (non-zero) columns on the right for $n > m$ or augmented with zero rows at the bottom for $m > n$

QR decomposition

Assume that $A \in \mathbb{R}^{m \times n}$ has rank r, with $r < m$ and $r < n$. The QR decomposition can be partitioned as

$$\underset{m \times n}{A} = \underset{m \times m}{Q} \cdot \underset{m \times n}{R} = \overset{m \times r}{[Q_1} \quad \overset{m \times (m-r)}{Q_2]} \cdot \begin{bmatrix} \underset{r \times r}{R_1} & \overset{r \times (n-r)}{R_2} \\ 0 & 0 \end{bmatrix}$$

Where:

- Q is an **orthogonal** matrix, $Q^\top Q = I_m$
- R_1 is **upper triangular** 

Let $\boldsymbol{b} \in \mathcal{C}(A) \iff \boldsymbol{b} = A\boldsymbol{x}$ for some $\boldsymbol{x} \iff \boldsymbol{b} = QR\boldsymbol{x} \iff \boldsymbol{b} = Q\boldsymbol{z}$ for some $\boldsymbol{z} = R\boldsymbol{x} \iff \boldsymbol{b} \in \mathcal{C}(Q)$

Then, the **columns** of Q are an **orthonormal basis** for $\mathcal{C}(A)$

QR decomposition

$$A = Q \cdot R = [Q_1 \quad Q_2] \cdot \begin{bmatrix} R_1 & R_2 \\ 0 & 0 \end{bmatrix}$$

Summarizing:

$$\begin{aligned} \text{Span}(A) &= \text{Span}(Q_1) \\ \text{Null}(A^\top) &= \text{Span}(Q_2) \\ \text{Span}(A^\top) &= \text{Span}(R_1^\top) \end{aligned}$$

The QR (RQ) factorizations **cannot be used** to determine the **rank** of a matrix in a reliable way. To reliably determine the rank, the **SVD** should be used

Outline

1. The four fundamental subspaces of a matrix
2. Projections
3. Singular Value Decomposition (SVD)
4. QR decomposition
5. **Rank condition and Cayley-Hamilton theorems**

Rank condition and Cayley-Hamilton theorems

Sylvester's inequality

Let A be a $p \times q$ matrix, and B a $q \times r$ matrix. Then:

$$\text{rank}(A) + \text{rank}(B) - q \leq \text{rank}(AB) \leq \min\{\text{rank}(A), \text{rank}(B)\}$$

For any $n \times n$ matrix A, the **characteristic polynomial** of A is defined as

$$a(\lambda) \coloneqq \det(\lambda I - A) = \lambda^n + a_1\lambda^{n-1} + a_2\lambda^{n-2} \dots + a_n$$

Cayley-Hamilton Theorem

Any square $n \times n$ matrix A satisfies its own characteristic equation, i.e

$$a(A) = A^n + a_1A^{n-1} + a_2A^{n-2} + \cdots + a_nI_n = \mathbf{0}$$

UNIVERSITÀ DEGLI STUDI DI BERGAMO | Dipartimento di Ingegneria Gestionale, dell'Informazione e della Produzione

ADAPTIVE LEARNING, ESTIMATION AND SUPERVISION OF DYNAMICAL SYSTEMS (ALES)

Lecture 6: Subspace identification – introduction and basic methods

Master Degree in COMPUTER ENGINEERING

Data Science and Data Engineering Curriculum

SPEAKER
Prof. Mirko Mazzoleni

PLACE
University of Bergamo

Syllabus

1. **Recursive and adaptive identification**
   - 1.1 Recursive ARX estimation (RLS)
   - 1.2 Least Mean Squares (LMS)
   - 1.3 Instrumental Variables (IV)
2. **Closed-loop identification**
3. **Subspace and MIMO identification**
   - 3.1 Singular Value Decomposition
   - 3.2 Impulse data: Ho-Kalman, Kung algorithms
   - 3.3 Generic I/O data: the MOESP algorithm
4. **Supervision of dynamical systems**
   - 4.1 Introduction to fault diagnosis
   - 4.2 Model-based fault diagnosis
   - 4.3 Parity space approaches
   - 4.4 Observer-based approaches
   - 4.5 Signal-based fault diagnosis
   - 4.6 Knowledge-based fault diagnosis

CASE STUDIES

- Virtual Reference Feedback Tuning
- Nuclear particles classification
- Leak detection in an industrial valve
- Bearing fault identification

Outline

1. Introduction to subspace identification
2. Review of the structural properties of a dynamical system
3. SISO identification with impulse data: noiseless case (Ho-Kalman algorithm)
4. SISO identification with impulse data: output noise case (Kung algorithm)
5. MIMO identification with impulse data

Outline

1. **Introduction to subspace identification**
2. Review of the structural properties of a dynamical system
3. SISO identification with impulse data: noiseless case (Ho-Kalman algorithm)
4. SISO identification with impulse data: output noise case (Kung algorithm)
5. MIMO identification with impulse data

Introduction to subspace identification

The classical **Prediction Error Method (PEM)** paradigm identifies **parametric INPUT-OUTPUT** transfer functions models $\hat{G}(z;\boldsymbol{\theta}), \hat{H}(z;\boldsymbol{\theta})$ based on several **user choices:**

1. A **model family** $\mathcal{M}(\boldsymbol{\theta})$
2. A **cost function** $J(\boldsymbol{\theta})$, like the sample variance of the one-step prediction error
3. The **optimization algorithm**, which might get stuck in **local minima**

If a **STATE-SPACE** model is required, the transfer functions can be converted in state-space by a procedure called **realization**

Introduction to subspace identification

State-space models are useful when:

- We want to deal with **MIMO systems**. In this case, the **input-output** polynomial notation becomes cumbersome, with an excess of model parameters
- We want to use the model for **advanced control design schemes**, as Linear Quadratic Regulator (LQR), Model Predictive Control (MPC)
- We want to **estimate the state variables** of the system, using observers (for deterministic systems) or Kalman filter (in the stochastic case)

Introduction to subspace identification

Multivariable systems (i.e., systems with many input and/or output signals) are often challenging to model, due to the **coupling** between different I/O channels.

It is sometimes advisable to **operate on subsets of the I/O channels**, aiming at the identification of **partial models:**

- the **more inputs** are considered, the more accurate the model will be, but **not all inputs may be equally important**
  if the model performance is not significantly deteriorated when an input is removed, then that input has small influence on the outputs and may be discarded
- the **more outputs** are considered, **the harder** the identification problem
  modeling one output at a time can reveal which outputs pose the greatest difficulty

Introduction to subspace identification

Subspace identification algorithms provide directly the **estimate of the system matrices** in a state-space representation

In the subspace framework:

1. We do not have to choose a model family
2. We do not have to choose a cost function
3. There is no optimizations involved
4. Transient signals (e.g. impulse responses) can be used

However, the **statistical analysis** of the estimates is much more **complicated**, since they are not based on an optimization procedure

Introduction to subspace identification

The subspace identification framework is purely **geometrical** and based on **linear algebra**

The basic idea of subspace methods is that, by **storing the input and output data in structured block Hankel matrices**, it is possible to retrieve certain subspaces that are related to the system matrices of the system

An example of such subspaces is the **column space** of the **observability matrix**

Key tools for subspace identification are:

1. The **QR** (or RQ) factorization
2. The **SVD**
3. A **linear least squares** problem

Outline

1. Introduction to subspace identification
2. **Review of the structural properties of a dynamical system**
3. SISO identification with impulse data: noiseless case (Ho-Kalman algorithm)
4. SISO identification with impulse data: output noise case (Kung algorithm)
5. MIMO identification with impulse data

State-space representation

Consider the **deterministic** SISO system in state-space form

$$\mathcal{S}: \begin{cases} \underset{n\times 1}{\boldsymbol{x}}(t+1) = \underset{n\times n}{A}\boldsymbol{x}(t) + \underset{n\times 1}{B}\underset{1\times 1}{u(t)} \\ \underset{1\times 1}{y(t)} = \underset{1\times n}{C}\boldsymbol{x}(t) + \underset{1\times 1}{D}u(t) \end{cases} \qquad \boldsymbol{x}(0)$$

The **transfer function** $G(z) = C(zI - A)^{-1}B + D$ can also be interpreted as the $\mathcal{Z}$-transform of the **impulse response** of the system (*aka* **Markov parameters**), $G(z) = \sum_{t=0}^{\infty} g(t)z^{-t}$

In the time domain the output $y(t)$ can be expressed as the **convolution** of the impulse response $g(t)$ with the input $u(t)$:

$$y(t) = \sum_{i=0}^{\infty} u(t-i)g(i) = g(0)u(t) + g(1)u(t-1) + \cdots$$

State-space representation

The sequence $\{g(0), g(1), g(2), \dots\}$ can be derived from $G(z)$ by **long division**, where

$$G(z) = \mathcal{Z}[g(t)] = \sum_{i=0}^{\infty} g(i)z^{-i}$$

Notice that $g(0) = 0$ if the system is **strictly proper**, i.e. if it has at least **one step delay** between $u(t)$ and $y(t)$, and so $D = 0$

Can we represent $g(t)$, and thus also $y(t)$, in terms of state-space matrices?

State-space representation

First, recall that the state representation (A, B, C, D) is **not unique**

Indeed, let T be a square matrix with $\det(T) \neq 0$, so that $\boldsymbol{x}_T(t) = T\boldsymbol{x}(t)$. Then, the state space system defined in terms of the matrices (A_T, B_T, C_T, D_T), where $A_T = TAT^{-1}$, $B_T = TB$, and $C_T = CT^{-1}$, $D_T = D$, has the **same input-output behavior** and the **same eigenvalues** as the original system (A, B, C, D).

In fact, there **exist infinite equivalent** state space realizations

Computing (A, B, C, D) from $G(z)$ is called **system realization.** A state space realization of $G(z)$ is called **minimal** if it has minimal state dimension n

System motion from state-space representation

Let us study the **forced motion** of system $\mathcal{S}$, starting from **null initial conditions**, $\boldsymbol{x}(0) = \mathbf{0}$

$$\boldsymbol{x}(0) = \mathbf{0}$$
$$\boldsymbol{x}(1) = A\boldsymbol{x}(0) + Bu(0) = Bu(0)$$
$$\boldsymbol{x}(2) = A\boldsymbol{x}(1) + Bu(1) = ABu(0) + Bu(1)$$
$$\vdots$$

$$y(0) = C\boldsymbol{x}(0) + Du(0) = Du(0)$$
$$y(1) = C\boldsymbol{x}(1) + Du(1) = CBu(0) + Du(1)$$
$$y(2) = C\boldsymbol{x}(2) + Du(2) = CABu(0) + CBu(1) + Du(2)$$
$$\vdots$$

In general, we obtain:

$$y(t) = \underbrace{D}_{g(0)} \cdot u(t) + \underbrace{\overset{1\times n}{C}\overset{n\times 1}{B}}_{g(1)} \cdot u(t-1) + \underbrace{\overset{1\times n}{C}\overset{n\times n}{A}\overset{n\times 1}{B}}_{g(2)} \cdot u(t-2) + \underbrace{CA^2B}_{g(3)} \cdot u(t-3) + \underbrace{CA^3B}_{g(4)} \cdot u(t-4) + \cdots$$

System motion from state-space representation

$$y(t) = \underbrace{D}_{g(0)} \cdot u(t) + \underbrace{CB}_{g(1)} \cdot u(t-1) + \underbrace{CAB}_{g(2)} \cdot u(t-2) + \underbrace{CA^2B}_{g(3)} \cdot u(t-3) + \underbrace{CA^3B}_{g(4)} \cdot u(t-4) + \cdots$$

Therefore, the system impulse response $g(t)$ can be represented as function of the state-space matrices as:

$$g(t) = \begin{cases} D & \text{if} \quad t = 0 \\ CA^{t-1}B & \text{if} \quad t > 0 \end{cases}$$

State-space representation

Observations:

- The initial state $\boldsymbol{x}(0)$ does not play any role in representing the impulse response $g(t)$ of $G(z)$ by state-space matrices

  This is due to the fact that model equivalence is considered in terms of transfer functions, and a transfer function only reflects the dynamic system response of systems that are initially at rest ($\boldsymbol{x}(0) = \mathbf{0}$)

- The relation

$$D = g(0)$$

  is immediate, and so the problem of constructing the matrix D can be separated from the construction of (A, B, C)

Observability

A system $\mathcal{S}$ is **observable** if there **do not exist** two different initial states such that the corresponding outputs are identical at all times t (when the same input is applied).

In other words, a system is observable if the information contained in the output $y(t)$ is **sufficient to reconstruct the state**

$\mathcal{S}$ is **observable if and only if** the **observability matrix** $\mathcal{O}_n \in \mathbb{R}^{n\times n}$ is **full rank**, i.e. $\text{rank}(\mathcal{O}_n) = n$

$$\underset{n\times n}{\mathcal{O}_n} = \begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{n-1} \end{bmatrix}$$

The matrix $\mathcal{O}_k = \begin{bmatrix} C^\top & (CA)^\top & (CA^2)^\top & \left(CA^{k-1}\right)^\top \end{bmatrix}^\top \in \mathbb{R}^{k\times n}$, $k > n$, is called **extended observability matrix**

Reachability

A system $\mathcal{S}$ is **reachable**, from an input $u(t)$, if starting from a **zero initial state**, one can always find an input signal that leads the system to any **given state in finite time**

Reachability from $u(t)$ can be interpreted as the fact that $u(t)$ affects **all the system states**

$\mathcal{S}$ is **reachable** (from $u(t)$) **if and only if** the **reachability matrix** $\mathcal{R}_n \in \mathbb{R}^{n\times n}$ is **full rank**, i.e. $\text{rank}(\mathcal{R}_n) = n$

$$\underset{n\times n}{\mathcal{R}_n} = [B \quad AB \quad A^2B \quad \cdots \quad A^{n-1}B]$$

The matrix $\mathcal{R}_k = [B \quad AB \quad A^2B \quad \cdots \quad A^{k-1}B] \in \mathbb{R}^{n\times k}$, $k > n$, is called **extended reachability matrix**

Minimal realization, observability and reachability

Proposition

A realization is **minimal if and only if** the corresponding system is **reachable** and **observable**

We will be interested in estimating **minimal realizations**

Outline

SISO identification with noiseless impulse data

Assume we collected a sequence of N **impulse response** measurements $y(t)$, in a **noise-free** situation, so that $y(t) = g(t)$

The impulse response might be a **very easy experiment** to perform, e.g. in mechanical or civil structures

Data:

- Input: $\{u(0) = 1,\ u(1) = 0,\ u(2) = 0,\ \dots,\ u(N) = 0\}$
- Output: $\{y(0) = g(0),\ y(1) = g(1),\ y(2) = g(2),\ \dots,\ y(N) = g(N)\}$

SISO identification with noiseless impulse data

In principle, it is possible to **transform** the samples $g(t)$ to obtain $G(z)$, and then computing a state-space **realization** of the system

However, this method is **hardly applicable** in practice, since most systems have an **infinite number** of Markov coefficients, so that

$$G(z) = \mathcal{Z}[g(t)] = \sum_{t=0}^{\infty} g(t) z^{-t}$$

cannot be computed

Hankel matrix

The **Hankel matrix** of order k of the system is defined as:

$$\underset{k \times k}{\mathcal{H}_k} = \begin{bmatrix} g(1) & g(2) & g(3) & \cdots & g(k) \\ g(2) & g(3) & g(4) & \cdots & g(k+1) \\ g(3) & g(4) & g(5) & \cdots & g(k+2) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ g(k) & g(k+1) & g(k+2) & \cdots & g(2k-1) \end{bmatrix}$$

We denote as $\mathcal{H}_{qd}$ the rectangular Hankel matrix with q rows and d columns

Hankel matrix decomposition

The Hankel matrix of order n (the exact system order) can be rewritten as follows:

$$\underset{n \times n}{\mathcal{H}_n} = \begin{bmatrix} CB & CAB & CA^2B & \cdots & CA^{n-1}B \\ CAB & CA^2B & CA^3B & \cdots & CA^nB \\ CA^2B & CA^3B & CA^4B & \cdots & CA^{n+1}B \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ CA^{n-1}B & CA^nB & CA^{n+1}B & \cdots & CA^{2n-2}B \end{bmatrix} = \underset{n \times n}{\begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{n-1} \end{bmatrix}} \cdot \underset{n \times n}{[B \;\; AB \;\; A^2B \;\; \cdots \;\; A^{n-1}B]} = \boxed{\mathcal{O} \cdot \mathcal{R}}$$

Since a **minimal realization** produces an **observable** and **reachable** system, it holds that

$$\text{rank}(\mathcal{O}) = n \qquad \textbf{and} \qquad \text{rank}(\mathcal{R}) = n$$

By the **Sylvester's inequality**, it follows that

$$\text{rank}(\mathcal{H}_n) = n$$

Hankel matrix decomposition

So, if we construct Hankel matrices of higher orders $k = n+1, n+2, \ldots$ the **rank** of $\mathcal{H}_k$ **will remain** n

This follows from the **Cayley-Hamilton theorem**. Recall that, for a square $n \times n$ matrix A,

$$\underset{n \times n}{a(A)} = \underset{n \times n}{A^n} + a_1 A^{n-1} + a_2 A^{n-2} \ldots + a_n I_n = \underset{n \times n}{\mathbf{0}}$$

Consider, for instance, the Hankel matrix $\underset{(n+k) \times (n+k)}{\mathcal{H}_{n+k}} = \underset{(n+k) \times n}{\mathcal{O}_{n+k}} \cdot \underset{n \times (n+k)}{\mathcal{R}_{n+k}}$ of order $n+k$, with $k \geq 0$. Then,

$$Ca(A)B = C(A^n + a_1 A^{n-1} + a_2 A^{n-2} \ldots + a_n I_n)B = \mathbf{0}$$

$$\Rightarrow CA^nB + a_1 CA^{n-1}B + a_2 CA^{n-2}B \ldots + a_n CB = \mathbf{0}$$

$$\Rightarrow CA^{n+k}B = -a_1 CA^{n-1+k}B - a_2 CA^{n-2+k}B \ldots - a_n CA^k B$$

Hankel matrix decomposition

$$CA^{n+k}B = -a_1CA^{n-1+k}B - a_2CA^{n-2+k}B \ldots - a_nCA^kB = \mathbf{0}$$

This shows that, from the $n+1$ row (and column) of $\mathcal{H}_{n+k}$, any row (or column) can be written **as linear combination** of the previous n rows (or column)

Thus, $\text{rank}(\mathcal{H}_{n+k})$ **will not increase** with increasing dimensions of $\mathcal{H}_{n+k}$

This implies that it is **not necessary to make any a priori assumption** on n in the identification process: the correct system order n can be found by **progressively constructing** bigger Hankel matrices

Hankel matrix decomposition

$\mathcal{H}_1 = [g(1)]$ ⇨ $\text{rank}(\mathcal{H}_1) = 1$

$\mathcal{H}_2 = \begin{bmatrix} g(1) & g(2) \\ g(2) & g(3) \end{bmatrix}$ ⇨ $\text{rank}(\mathcal{H}_2) = 2$

$\mathcal{H}_3 = \begin{bmatrix} g(1) & g(2) & g(3) \\ g(2) & g(3) & g(4) \\ g(3) & g(4) & g(5) \end{bmatrix}$ ⇨ $\text{rank}(\mathcal{H}_3) = 3$

⋮

$\mathcal{H}_n$ ⇨ $\text{rank}(\mathcal{H}_n) = n$

$\mathcal{H}_{n+1}$ ⇨ $\text{rank}(\mathcal{H}_{n+1}) = n$

- Augmenting further the Hankel matrix does not increase the rank
- Then, the system order equals n
- Notice that the rank of the rectangular Hankel matrix $\mathcal{H}_{ij}$ is still n if both $i, j > n$

Estimate system matrices from the Hankel matrix

As we saw, it is possible to factorize $\mathcal{H}_{n+1} \in \mathbb{R}^{(n+1)\times(n+1)}$ as the product of the extended observability matrix $\mathcal{O}_{n+1}$ and the extended reachability matrix $\mathcal{R}_{n+1}$:

$$\underset{(n+1)\times(n+1)}{\mathcal{H}_{n+1}} = \underset{(n+1)\times n}{\mathcal{O}_{n+1}} \cdot \underset{n\times(n+1)}{\mathcal{R}_{n+1}}$$

We can use the **SVD** to estimate (A, B, C, D) up to **similarity transformation** T where $\underset{n\times n}{\boldsymbol{x}_T(t)} = T\boldsymbol{x}(t)$, that is $(A_T, B_T, C_T, D_T) = (TAT^{-1}, TB, CT^{-1}, D)$. Let

$$\underset{(n+1)\times(n+1)}{\mathcal{H}_{n+1}} = \underset{(n+1)\times n}{U_n} \overset{n\times n}{\Sigma_n} \underset{n\times(n+1)}{V_n^\top}$$

with $\Sigma_n \in \mathbb{R}^{n\times n}$ and $\text{rank}(\Sigma_n) = n$

Estimate system matrices from the Hankel matrix

Then, U_n is a basis for $\mathcal{C}(\mathcal{H}_{n+1})$ and for $\mathcal{C}(\mathcal{O}_{n+1})$. Also, the columns of U_n belong to $\mathcal{C}(\mathcal{H}_{n+1})$ and to $\mathcal{C}(\mathcal{O}_{n+1})$. Thus, we can express U_n as

$$\underset{(n+1)\times n}{U_n} = \underset{(n+1)\times n}{\mathcal{O}_{n+1}} \overset{n\times n}{T^{-1}} = \begin{bmatrix} CT^{-1} \\ CT^{-1}(TAT^{-1}) \\ \vdots \\ CT^{-1}(TAT^{-1})^{n-1} \end{bmatrix} = \begin{bmatrix} C_T \\ C_TA_T \\ \vdots \\ C_TA_T^{n-1} \end{bmatrix}$$

We can set:

$$\underset{(n+1)\times n}{\hat{\mathcal{O}}_{n+1}} = \underset{(n+1)\times n}{U_n} \cdot \underset{n\times n}{\Sigma^{1/2}}$$

$$\underset{n\times(n+1)}{\hat{\mathcal{R}}_{n+1}} = \Sigma^{1/2} \cdot \underset{n\times(n+1)}{V_n^\top}$$

⇨ $\mathcal{H}_{n+1} = \hat{\mathcal{O}}_{n+1} \cdot \hat{\mathcal{R}}_{n+1} = U_n\Sigma^{1/2} \cdot \Sigma^{1/2}V_n^\top$

Estimate system matrices from the Hankel matrix

Once we have $\hat{\mathcal{O}}_{n+1}$ and $\hat{\mathcal{R}}_{n+1}$, the system matrices B and C can be estimated as follows:

$\underset{1\times n}{\hat{C}} = \underset{(n+1)\times n}{\hat{\mathcal{O}}_{n+1}(1,:)}$ ⇨ First row of $\mathcal{O}_{n+1}$

$\underset{n\times 1}{\hat{B}} = \underset{n\times(n+1)}{\hat{\mathcal{R}}_{n+1}(:,1)}$ ⇨ First column of $\mathcal{R}_{n+1}$

As for $\hat{A}$, observe that

$$\underset{(n+1)\times n}{\hat{\mathcal{O}}_{n+1}} = \begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{n-1} \\ CA^n \end{bmatrix} = \begin{bmatrix} \mathcal{O}_n \\ CA^n \end{bmatrix} \quad \text{and} \quad \underset{(n+1)\times n}{\hat{\mathcal{O}}_{n+1}A} = \begin{bmatrix} CA \\ CA^2 \\ \vdots \\ CA^n \\ CA^{n+1} \end{bmatrix}$$

$\hat{\mathcal{O}}_{n+1}(2:n+1,:)$ $\quad$ $\hat{\mathcal{O}}_{n+1}(1:n,:)A$

Estimate system matrices from the Hankel matrix

Then, it holds that (*shift invariance property*):

$$\underset{n\times n}{\hat{\mathcal{O}}_{n+1}(1:n,:)}\ \underset{n\times n}{A} = \underset{n\times n}{\hat{\mathcal{O}}_{n+1}(2:n+1,:)}$$

where, by construction, $\hat{\mathcal{O}}_{n+1}(1:n,:)$ is a $n \times n$ full rank matrix, and thus invertible

Therefore, we can estimate A as follows: $\hat{A} = \hat{\mathcal{O}}_{n+1}(1:n,:)^{-1} \cdot \hat{\mathcal{O}}_{n+1}(2:n+1,:)$

Notice that the system has been estimated:

- using **only impulse response data**
- **without any a priori knowledge** or assumption on the system
- **without any optimization process**, just plain linear algebra

Outline

SISO identification with noisy impulse data

The Ho-Kalman procedure was known since the '60s but is **hardly applicable to real problems**, where gross estimation errors are typically experienced

In fact, various issues prevent the method from functioning as anticipated:

- **measurement disturbances** that corrupt the impulse response data
- **numerical approximation** (e.g., rounding) affecting the data

We now assume **noisy** impulse response **measurements** $\tilde{g}(t) = g(t) + e(t)$

Data:

- Input: $\{u(0) = 1,\ u(1) = 0,\ u(2) = 0,\ \dots,\ u(N) = 0\}$
- Output: $\{y(0) = \tilde{g}(0),\ y(1) = \tilde{g}(1),\ y(2) = \tilde{g}(2),\ \dots,\ y(N) = \tilde{g}(N)\}$

Rectangular Hankel matrix

With **noisy data**, the previous procedure for estimating the system order using the **rank of the Hankel matrix is no more reliable**. A solution is to rely on the SVD, and in particular looking at the singular values

First, construct the **rectangular** Hankel matrix $\widetilde{\mathcal{H}}_{qd}$, using all the N available noisy data

$$\underset{q\times d}{\widetilde{\mathcal{H}}_{qd}} = \begin{bmatrix} \tilde{g}(1) & \tilde{g}(2) & \tilde{g}(3) & \cdots & \tilde{g}(d) \\ \tilde{g}(2) & \tilde{g}(3) & \tilde{g}(4) & \cdots & \tilde{g}(d+1) \\ \tilde{g}(3) & \tilde{g}(4) & \tilde{g}(5) & \cdots & \tilde{g}(d+2) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \tilde{g}(q) & \tilde{g}(q+1) & \tilde{g}(q+2) & \cdots & \tilde{g}(q+d-1) \end{bmatrix}$$

so that $q + d - 1 = N$

Rectangular Hankel matrix

How to choose q and d? Usually:

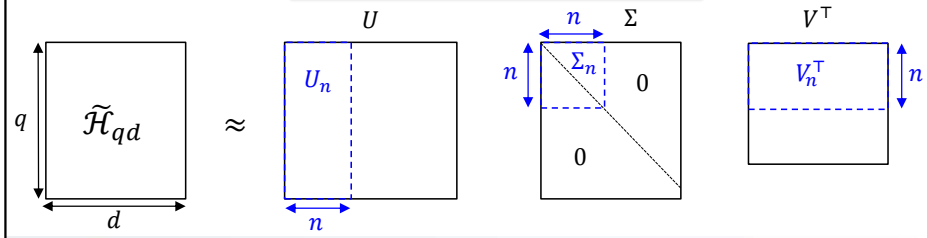
- $q \leq d$
- In general $q \approx \frac{1}{2}d$. To estimate the observability matrix it is convenient to construct a «fat» Hankel matrix ($q \ll d$)
- From a computational point of view, it is better (less burden) taking q small. However, a too small q compromises the results

For instance, if $N = 1000$, we can take $q \approx 300$ and $d \approx 700$. In any case, q and d must be larger than n

Hankel matrix with noisy data

The idea is to **approximate** the generic $\widetilde{\mathcal{H}}_{qd}$ Hankel matrix with a (numerical) **rank-n approximation**, where n is chosen by looking at the singular values of a **SVD of** $\widetilde{\mathcal{H}}_{qd}$

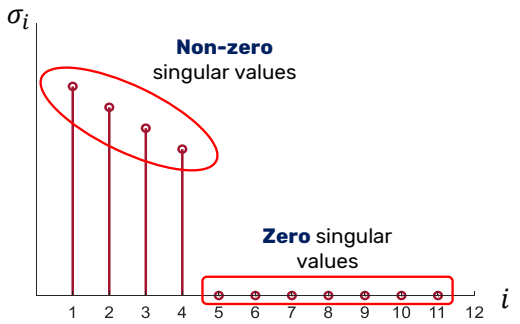
$$\underset{q\times d}{\widetilde{\mathcal{H}}_{qd}} \approx \underset{q\times n}{U_n} \cdot \underset{n\times n}{\Sigma_n} \cdot \underset{n\times d}{V_n^\top} := \underset{q\times d}{\hat{\mathcal{H}}_{qd}}$$



SISO identification with noisy impulse data

In absence of noise, the **system order** exactly coincides with **the rank of the Hankel matrix**, which can be exactly evaluated as the **number of non-zero singular values**

When the data are affected by **noise**, the rank of the Hankel matrix is a **gross overestimate** of the system order



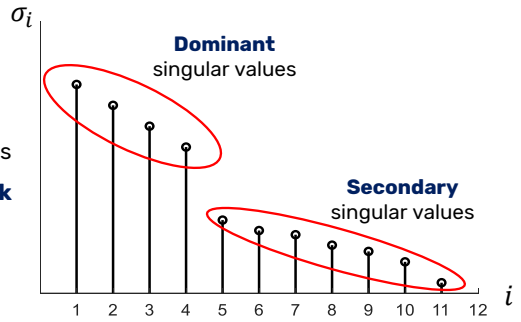
SISO identification with noisy impulse data

In practice, one must classify the singular values in two categories:

- **dominant** singular values
- **secondary** singular values

according to their **relative differences**

The number of dominant singular values defines the matrix **numerical rank** (usually lower than the rank)



Hankel matrix with noisy data

Contrary to $\mathcal{H}_{qd}$, whose rank is generally full ($\text{rank}(\mathcal{H}_{qd}) = \min(q,d)$), the rank of the matrix $\widehat{\mathcal{H}}_{qd} = U_n \Sigma_n V_n^\top$ **equals exactly the system order** n

- The elements of $\widehat{\mathcal{H}}_{qd} = U_n \Sigma_n V_n^\top$ can be interpreted as the **samples of the true**, noise-free **impulse response** of the system
- The elements of $\mathcal{H}_{\text{res}} = \tilde{\mathcal{H}}_{qd} - \widehat{\mathcal{H}}_{qd}$ can be interpreted as the **noise contribution** to impulse measurements

Hankel matrix with noisy data

Now, let

$$\underset{n\times n}{\Sigma_n^{1/2}} = \begin{bmatrix} \sqrt{\sigma_1} & 0 & \cdots & 0 \\ 0 & \sqrt{\sigma_2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sqrt{\sigma_n} \end{bmatrix}$$

We can estimate the (extended) observability and reachability matrices as

$$\widehat{\mathcal{H}}_{qd} = U_n \Sigma_n V_n^\top = \underset{q\times n}{\left(U_n \underset{n\times n}{\Sigma_n^{1/2}}\right)} \cdot \underset{n\times d}{\left(\Sigma_n^{1/2} V_n^\top\right)} = \underset{q\times n}{\hat{\mathcal{O}}_q} \cdot \underset{n\times d}{\hat{\mathcal{R}}_d}$$

Estimated extended observability matrix of order q

Estimated extended reachability matrix of order d

Hankel matrix with noisy data

Finally, estimate (A, B, C, D) from $\hat{\mathcal{O}}_q$ and $\hat{\mathcal{R}}_d$ as explained previously

$\underset{1\times n}{\hat{C}} = \underset{q\times n}{\hat{\mathcal{O}}_q}(1,:)$ ⇨ First row of $\hat{\mathcal{O}}_q$

$\underset{n\times 1}{\hat{B}} = \underset{n\times d}{\hat{\mathcal{R}}_d}(:,1)$ ⇨ First column of $\hat{\mathcal{R}}_d$

The shift invariance property is exploited to estimate A: $\underset{(q-1)\times n}{\hat{\mathcal{O}}_q(1:q-1,:)} \underset{n\times n}{A} = \underset{(q-1)\times n}{\hat{\mathcal{O}}_q(2:q,:)}$

Since $\hat{\mathcal{O}}_q(1:q-1,:)$ is not square we have to use the **pseudoinverse** to solve for A. We first left-multiply both sides by $\hat{\mathcal{O}}_q(1:q-1,:)^\dagger$, obtaining:

$$\underset{n\times n}{\hat{A}} = \underset{n\times(q-1)}{\hat{\mathcal{O}}_q(1:q-1,:)^\dagger} \cdot \underset{(q-1)\times n}{\hat{\mathcal{O}}_q(2:q,:)}$$

Estimate system matrices from the Hankel matrix

Kung's realization algorithm

- Collect noisy **impulse response** data
  - ✓ Input: $\{u(0)=1,\ u(1)=0, \dots,\ u(N)=0\}$
  - ✓ Output: $\{y(0)=\tilde{g}(0),\ y(1)=\tilde{g}(1), \dots,\ y(N)=\tilde{g}(N)\}$
- Construct Hankel matrix $\tilde{\mathcal{H}}_{qd}$, with $q \approx \frac{1}{2}d$ and $N = q + d - 1$
- Perform SVD on $\tilde{\mathcal{H}}_{qd} = U\Sigma V^{\top}$
- Choose n and set $U_n = U(:, 1:n), \quad \Sigma_n = \Sigma(1:n, 1:n), \quad V_n^{\top} = V^{\top}(1:n, :)$
- Set $\hat{\mathcal{O}}_q = U_n \Sigma_n^{1/2}, \quad \hat{\mathcal{R}}_d = \Sigma_n^{1/2} V_n^{\top}$
- Estimate $\hat{C} = \hat{\mathcal{O}}_q(1, :), \quad \hat{B} = \hat{\mathcal{R}}_d(:, 1), \quad \hat{D} = \tilde{g}(0)$
- Estimate $\hat{A} = \hat{\mathcal{O}}_q(1: q-1, :)^{\dagger} \cdot \hat{\mathcal{O}}_q(2: q, :)$

Example (SISO Kung's algorithm)

Consider the following $\mathrm{ARX}(n_a = 2, n_b = 1, k = 1)$ system:

$$S: y(t) = a_1 y(t-1) + a_2 y(t-2) + b_1 u(t-1) + b_2 u(t-2) + e(t)$$

where $a_1 = 0.2, a_2 = 0.35, b_1 = 1, b_2 = -5$ and $e(t) \sim \mathrm{WGN}(0, 0.2^2)$

Generate $N = 30$ samples of the impulse response and construct the Hankel matrix $\mathcal{H}_{qd}$, after having chosen $q = 10$ and $d = 20$

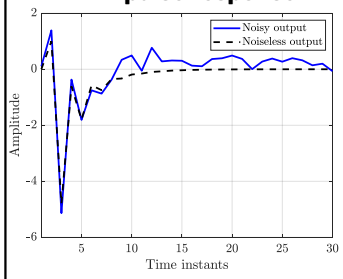
Perform the SVD obtaining the required U, Σ, and V matrices

Inspecting the singular values, it is relatively easy to decide that the model order should be set to 2 and reduce Σ accordingly to its 2×2 upper left block

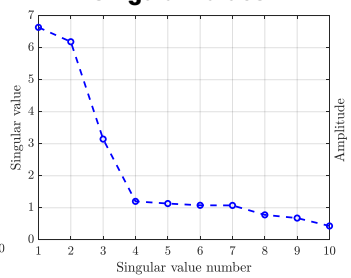
Finally, we use matrices U_n, Σ_n and V_n to compute $\hat{\mathcal{O}}_q$ and $\hat{\mathcal{R}}_d$, and then, in turn, employ the latter matrices to construct the estimates of (A, B, C, D)

Example (SISO Kung's algorithm)

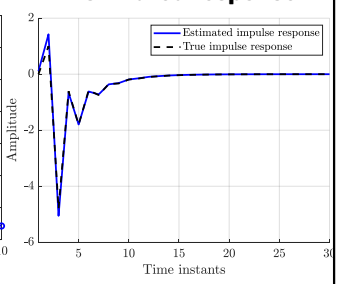
Impulse response



Singular values



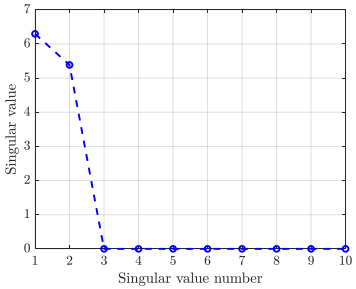
Estimated response

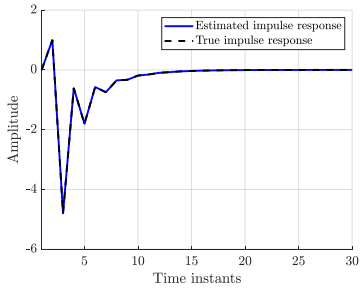


	Zeros	Poles		Gain
True parameters	5	-0.5	0.7	-8.8
Estimated parameters	3.115	-0.470	0.693	-8.64

Example (SISO Kung's algorithm)

To verify the correctness of the method, repeat the experiment in **the absence of noise**





In this case, **only the first two singular values are different from zero**, so that the reduction of matrix Σ to its upper left block **does not result in a loss of information**

Accordingly, the **model estimation is perfect**

Outline

1. Introduction to subspace identification
2. Review of the structural properties of a dynamical system
3. SISO identification with impulse data: noiseless case (Ho-Kalman algorithm)
4. SISO identification with impulse data: output noise case (Kung algorithm)
5. **MIMO identification with impulse data**

MIMO identification with impulse data

Consider the MIMO system:

$$\mathcal{S}: \begin{cases} \underset{n\times 1}{\boldsymbol{x}}(t+1) = \underset{n\times n}{A}\boldsymbol{x}(t) + \underset{n\times m_u}{B}\underset{m_u\times 1}{\boldsymbol{u}}(t) \\ \underset{p\times 1}{\boldsymbol{y}}(t) = \underset{p\times n}{C}\boldsymbol{x}(t) + \underset{p\times m_u}{D}\boldsymbol{u}(t) \end{cases} \qquad \boldsymbol{x}(0)$$

The method carries over **seamlessly to the MIMO case**, albeit with a heavier notation

In particular, notice that the terms CB, CAB, CA^2B, … that enter in the Hankel matrix **are not scalars anymore**, but $p \times m_u$ **matrices**

Thus, the (i, j) elements represents the response of the i**-th output to the** j**-th input**

MIMO identification with impulse data

We need to express the Markov parameters as $p \times m_u$ matrices:

$$\underset{p\times m_u}{W(t)} = \begin{cases} \underset{p\times m_u}{D} & \text{if } t = 0 \\ \underset{p\times n}{C}\underset{n\times n}{A^{t-1}}\underset{n\times m_u}{B} & \text{if } t > 0 \end{cases}$$

and the Hankel matrix is actually a ***block* Hankel matrix**

After the construction of $\widetilde{\mathcal{H}}_{qd}$ using $W(t)$, apply the SVD and subset the resulting matrices accordingly

Block Hankel matrix

For instance, the Hankel matrix of a 2×2 MIMO system will look as follows, where $g_{ij}(t)$ is the impulse response of the i-th output to the j-th input

$$\underset{\substack{(qp)\times(dm_u)\\ (q2)\times(d2)}}{\mathcal{H}_{qd}} = \underset{2\times 2}{\begin{bmatrix} W(1) & W(2) & \cdots & W(d) \\ W(2) & W(3) & \cdots & W(d+1) \\ \vdots & \vdots & \ddots & \vdots \\ W(q) & W(q+1) & \cdots & W(q+d-1) \end{bmatrix}} =$$

$$= \begin{bmatrix} g_{11}(1) & g_{12}(1) & g_{11}(2) & g_{12}(2) & \cdots & g_{11}(d) & g_{12}(d) \\ g_{21}(1) & g_{22}(1) & g_{21}(2) & g_{22}(2) & \cdots & g_{21}(d) & g_{22}(d) \\ g_{11}(2) & g_{12}(2) & & & & g_{11}(d+1) & g_{12}(d+1) \\ g_{21}(2) & g_{22}(2) & & & & g_{21}(d+1) & g_{22}(d+1) \\ \vdots & \vdots & & & \ddots & \vdots & \\ g_{11}(q) & g_{12}(q) & g_{11}(q+1) & g_{12}(q+1) & \cdots & g_{11}(q+d-1) & g_{12}(q+d-1) \\ g_{21}(q) & g_{22}(q) & g_{21}(q+1) & g_{22}(q+1) & \cdots & g_{21}(q+d-1) & g_{22}(q+d-1) \end{bmatrix}$$

MIMO identification with impulse data

The observability matrix $\hat{\mathcal{O}}_q \in \mathbb{R}^{(q\cdot p)\times n}$ and the reachability matrix $\hat{\mathcal{R}}_d \in \mathbb{R}^{n\times(d\cdot m_u)}$ are **estimated as in the SISO case**, by using the SVD to obtain $U \in \mathbb{R}^{(q\cdot p)\times(q\cdot p)}, \Sigma \in \mathbb{R}^{(q\cdot p)\times(d\cdot m_u)}$ and $V^\top \in \mathbb{R}^{(d\cdot m_u)\times(d\cdot m_u)}$

$$\underset{(qp)\times(dm_u)}{\tilde{\mathcal{H}}_{qd}} \approx \underset{(qp)\times(qp)}{U} \cdot \Sigma \cdot \underset{(dm_u)\times(dm_u)}{V^\top}$$

And then performing a rank-n approximation

$$\underset{(qp)\times(dm_u)}{\tilde{\mathcal{H}}_{qd}} = \underset{(qp)\times n}{U_n} \cdot \underset{n\times n}{\Sigma_n} \cdot \underset{n\times md_u}{V_n^\top} = \left(U_n \Sigma_n{}^{1/2}\right) \cdot \left(\Sigma_n^{1/2} V_n^\top\right) = \underset{(qp)\times n}{\hat{\mathcal{O}}_q} \cdot \underset{n\times(dm_u)}{\hat{\mathcal{R}}_d}$$

MIMO identification with impulse data

Hence,

$$\underset{p\times n}{\hat{C}} = \hat{\mathcal{O}}_q(1{:}\,p,:) \qquad \underset{p\times m_u}{\hat{D}} = W(0)$$

$$\underset{n\times m_u}{\hat{B}} = \hat{\mathcal{R}}_d(:,1{:}\,m_u)$$

$$\underset{n\times n}{\hat{A}} = \hat{\mathcal{O}}_q(1{:}\,q-p,:)^\dagger \cdot \hat{\mathcal{O}}_q(p+1{:}\,q,:)$$

$$\underset{n\times(q-p)}{\qquad} \qquad \underset{(q-p)\times n}{\qquad}$$

Remark:

We could have used also $\hat{\mathcal{O}}_q = U_n$. In this case, we estimate a different observability matrix, up to a linear transformation T of the states so that $\boldsymbol{x}'(t) = T\boldsymbol{x}(t)$ and thus $\mathcal{O}'_q = \mathcal{O}_q T^{-1}$. However, the importat thing is that $\mathcal{C}(\mathcal{O}_q) = \mathcal{C}(\mathcal{O}'_q)$

Example (MIMO Kung's algorithm)

Consider the system:

$$\mathcal{S}: \begin{cases} \boldsymbol{x}(t+1) = A\boldsymbol{x}(t) + B\boldsymbol{u}(t) \\ \boldsymbol{y}(t) = C\boldsymbol{x}(t) + \boldsymbol{e}(t) \end{cases} \quad A = \begin{bmatrix} 0.7 & 0 \\ 0 & 0.3 \end{bmatrix} \quad B = \begin{bmatrix} -0.1 & 0.2 & 0.6 \\ 0.9 & -0.5 & -0.4 \end{bmatrix} \quad C = \begin{bmatrix} 0.5 & 0.2 \\ 0.6 & -0.8 \end{bmatrix}$$

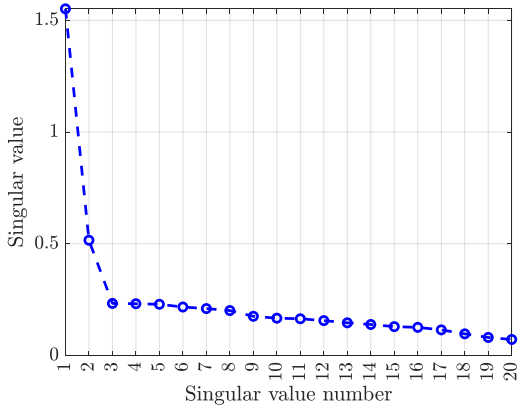
and $\boldsymbol{e}(t) \sim \mathrm{WGN}(0, \sigma^2 I_2)$, with $\sigma = 0.02$. Notice that the system matrices require **14 parameters** as opposed to the **24 necessary** to specify the 6 transfer functions

The system has $p = 2$ outputs, $m_u = 3$ inputs and $n = 2$ states. Observe $N = 30$ data. The Hankel matrix is constructed with $q = 10, d = 20$

The impulse response of the system is collected by **activating one input at a time** and observing the 2 corresponding outputs

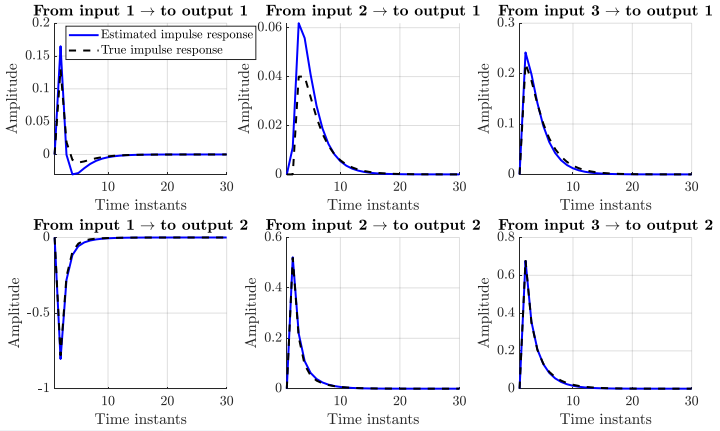
Example (MIMO Kung's algorithm)

The analysis of its singular values suggests that the order of the system is $n = 2$



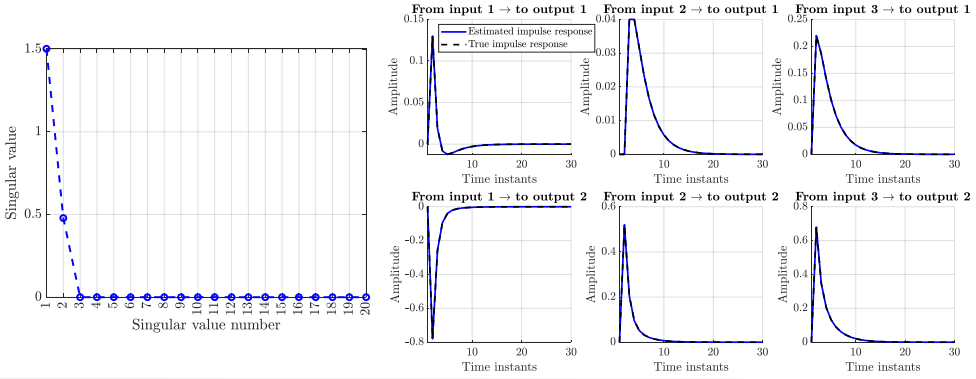
Example (MIMO Kung's algorithm)

Let's compare the true and estimated impulse responses



Example (MIMO Kung's algorithm)

To check the correctness of the method, we re-do the experiment with **no output-noise**



UNIVERSITÀ DEGLI STUDI DI BERGAMO | Dipartimento di Ingegneria Gestionale, dell'Informazione e della Produzione

UNIVERSITÀ DEGLI STUDI DI BERGAMO | Dipartimento di Ingegneria Gestionale, dell'Informazione e della Produzione

ADAPTIVE LEARNING, ESTIMATION AND SUPERVISION OF DYNAMICAL SYSTEMS (ALES)

Lecture 7: Multivariable Output Error State Space (MOESP) subspace identification - part I

Master Degree in COMPUTER ENGINEERING

Data Science and Data Engineering Curriculum

SPEAKER
Prof. Mirko Mazzoleni

PLACE
University of Bergamo

Syllabus

1. **Recursive and adaptive identification**
   - 1.1 Recursive ARX estimation (RLS)
   - 1.2 Least Mean Squares (LMS)
   - 1.3 Instrumental Variables (IV)
2. **Closed-loop identification**
3. **Subspace and MIMO identification**
   - 3.1 Singular Value Decomposition
   - 3.2 Impulse data: Ho-Kalman, Kung algorithms
   - 3.3 Generic I/O data: the MOESP algorithm
4. **Supervision of dynamical systems**
   - 4.1 Introduction to fault diagnosis
   - 4.2 Model-based fault diagnosis
   - 4.3 Parity space approaches
   - 4.4 Observer-based approaches
   - 4.5 Signal-based fault diagnosis
   - 4.6 Knowledge-based fault diagnosis

CASE STUDIES

- Virtual Reference Feedback Tuning
- Nuclear particles classification
- Leak detection in an industrial valve
- Bearing fault identification

Outline

1. MIMO subspace identification with generic noiseless I\O data: MOESP
2. Estimating (A, C)
3. Improving the computational efficiency with RQ factorization
4. Estimating $(B, D, x(0))$ given $(\hat{A}, \hat{C})$

Outline

1. **MIMO subspace identification with generic noiseless I\O data: MOESP**
2. Estimating (A, C)
3. Improving the computational efficiency with RQ factorization
4. Estimating $(B, D, x(0))$ given $(\hat{A}, \hat{C})$

MOESP algorithm (orthogonal projection algorithm)

Consider the **deterministic** MIMO system

$$\mathcal{S}: \begin{cases} \underset{n\times 1}{\boldsymbol{x}(t+1)} = \underset{n\times n}{A}\boldsymbol{x}(t) + \underset{n\times m_u}{B}\underset{m_u\times 1}{\boldsymbol{u}(t)} \\ \underset{p\times 1}{\boldsymbol{y}(t)} = \underset{p\times n}{C}\boldsymbol{x}(t) + \underset{p\times m_u}{D}\underset{m_u\times 1}{\boldsymbol{u}(t)} \end{cases} \qquad \boldsymbol{x}(0)$$

The estimation of matrices (A, B, C, D) is divided in two steps:

1. Estimate (C, A) through estimation of the **extended observability matrix** $\mathcal{O}_s$, $s > n$
2. Estimate (B, D) via a **least squares** problem

Outline

1. MIMO subspace identification with generic noiseless I\O data: MOESP
2. **Estimating** (A, C)
3. Improving the computational efficiency with RQ factorization
4. Estimating $(B, D, x(0))$ given $(\hat{A}, \hat{C})$

The data equation

The response to a **generic input** (not limited to impulse input) up to time t over a **window of finite length** s can be written as follows:

$$\boldsymbol{y}(t+s-1) = C\boldsymbol{x}(t+s-1) + D\boldsymbol{u}(t+s-1)$$

$$= C \cdot A\boldsymbol{x}(t+s-2) + C \cdot B\boldsymbol{u}(t+s-2) + D\boldsymbol{u}(t+s-1)$$

$$= CA^2 \cdot \boldsymbol{x}(t+s-3) + CAB \cdot \boldsymbol{u}(t+s-3) + CB \cdot \boldsymbol{u}(t+s-2) + D\boldsymbol{u}(t+s-1)$$

$$= CA^{s-1}\boldsymbol{x}(t) + CA^{s-2}B \cdot \boldsymbol{u}(t) + CA^{s-3}B \cdot \boldsymbol{u}(t+1) + \cdots + CB \cdot \boldsymbol{u}(t+s-2) + D\boldsymbol{u}(t+s-1)$$

In general, the output at a **generic time instant** t **from the beginning** is

$$\boldsymbol{y}(t) = \sum_{\tau=0}^{t-1} CA^{t-\tau-1}B \cdot \boldsymbol{u}(\tau) + D\boldsymbol{u}(t) + CA^t\boldsymbol{x}(0)$$

The data equation

The previous expression corresponds to the following matrix equation:

$$\underbrace{\begin{bmatrix} \boldsymbol{y}(t) \\ \boldsymbol{y}(t+1) \\ \boldsymbol{y}(t+2) \\ \vdots \\ \boldsymbol{y}(t+s-1) \end{bmatrix}}_{\overrightarrow{\boldsymbol{y}_s}(t)} = \underbrace{\begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{s-1} \end{bmatrix}}_{\mathcal{O}_s} \cdot \boldsymbol{x}(t) + \underbrace{\begin{bmatrix} D & 0 & 0 & \cdots & 0 \\ CB & D & 0 & \cdots & 0 \\ CAB & CB & D & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ CA^{s-2}B & CA^{s-3}B & \cdots & CB & D \end{bmatrix}}_{\mathcal{T}_s} \cdot \underbrace{\begin{bmatrix} \boldsymbol{u}(t) \\ \boldsymbol{u}(t+1) \\ \boldsymbol{u}(t+2) \\ \vdots \\ \boldsymbol{u}(t+s-1) \end{bmatrix}}_{\overrightarrow{\boldsymbol{u}_s}(t)}$$

Or, more systematically, to:

$$\overrightarrow{\boldsymbol{y}_s}(t) = \mathcal{O}_s \cdot \boldsymbol{x}(t) + \mathcal{T}_s \cdot \overrightarrow{\boldsymbol{u}_s}(t)$$

The data equation

Replicate the motion equations at **different time instants:**

$$\mathbf{Y} \coloneqq \underset{ps \times (N-s+1)}{\left[\overrightarrow{\boldsymbol{y}_s}(1) \quad \overrightarrow{\boldsymbol{y}_s}(2) \quad \cdots \quad \overrightarrow{\boldsymbol{y}_s}(N-s+1)\right]} = \begin{bmatrix} \boldsymbol{y}(1) & \boldsymbol{y}(2) & \cdots & \boldsymbol{y}(N-s+1) \\ \boldsymbol{y}(2) & \boldsymbol{y}(3) & \cdots & \boldsymbol{y}(N-s+2) \\ \vdots & \vdots & \cdots & \vdots \\ \boldsymbol{y}(s) & \boldsymbol{y}(s+1) & \cdots & \boldsymbol{y}(N) \end{bmatrix}$$

$\overrightarrow{\boldsymbol{y}_s}(t)$ starts from t and it ends at $t+s-1$

$$\mathbf{U} \coloneqq \underset{m_u s \times (N-s+1)}{\left[\overrightarrow{\boldsymbol{u}_s}(1) \quad \overrightarrow{\boldsymbol{u}_s}(2) \quad \cdots \quad \overrightarrow{\boldsymbol{u}_s}(N-s+1)\right]} = \begin{bmatrix} \boldsymbol{u}(1) & \boldsymbol{u}(2) & \cdots & \boldsymbol{u}(N-s+1) \\ \boldsymbol{u}(2) & \boldsymbol{u}(3) & \cdots & \boldsymbol{u}(N-s+2) \\ \vdots & \vdots & \cdots & \vdots \\ \boldsymbol{u}(s) & \boldsymbol{u}(s+1) & \cdots & \boldsymbol{u}(N) \end{bmatrix}$$

$$\mathbf{X} \coloneqq \underset{n \times (N-s+1)}{\left[\boldsymbol{x}(1) \quad \boldsymbol{x}(2) \quad \cdots \quad \boldsymbol{x}(N-s+1)\right]}$$

The data equation

It follows the following **data equation**

$$\underset{ps\times(N-s+1)}{\mathbf{Y}} = \underset{ps\times n}{\mathcal{O}_s}\cdot\overset{n\times(N-s+1)}{\mathbf{X}} + \underset{ps\times m_u s}{\mathcal{T}_s}\cdot\overset{m_u s\times(N-s+1)}{\mathbf{U}}$$

where in general $n < s \ll N$

The data equation relates **matrices constructed from the data** to **matrices constructed from the system matrices**

This representation will allow to derive information on the system matrices (A, B, C, D) from data matrices

Removing the input terms

The equation

$$\underset{ps\times h}{\mathbf{Y}} = \underset{ps\times n}{\mathcal{O}_s}\cdot\underset{n\times h}{\mathbf{X}} + \underset{ps\times m_u s}{\mathcal{T}_s}\cdot\underset{m_u s\times h}{\mathbf{U}}$$

- $h \coloneqq N - s + 1$

shows that each **column** of $\mathbf{Y}$ is a linear combination of **columns** of $\mathcal{O}_s$ and $\mathcal{T}_s$. Alternatively, the **rows** of $\mathbf{Y}$ are a linear combination of **rows** of $\mathbf{X}$ and $\mathbf{U}$

The term $\mathcal{T}_s\mathbf{U}$ can be eliminated by right multiplying all terms by the projection matrix $\mathbf{U}^\perp$

$$\underset{h\times h}{\mathbf{U}^\perp} = I_h - \underset{h\times m_u s}{\mathbf{U}^\top}\underset{m_u s\times m_u s}{(\mathbf{U}\mathbf{U}^\top)^{-1}}\underset{m_u s\times h}{\mathbf{U}}$$

where it has been assumed that $\mathbf{U}\mathbf{U}^\top$ is **invertible**, i.e it has full row rank $m_u s$ (which is equivalent to imposing a **persistent excitation** condition on $u(\cdot)$)

Orthogonal projection

Recall that, given $A \in \mathbb{R}^{m\times n}$

If $\operatorname{rank}(A) = n$

$$\underset{m\times m}{P_{\text{col}}} \coloneqq \underset{m\times n}{A}\underset{n\times n}{(A^\top A)^{-1}}\underset{n\times m}{A^\top} = \text{projection matrix onto the } \textbf{column space} \text{ of } A, \text{ i.e. } \mathcal{C}(A)$$

If $\operatorname{rank}(A) = m$

$$\underset{n\times n}{P_{\text{row}}} \coloneqq \underset{n\times m}{A^\top}\underset{m\times m}{(AA^\top)^{-1}}\underset{m\times n}{A} = \text{projection matrix onto the } \textbf{row space} \text{ of } A, \text{ i.e. } \mathcal{C}(A^\top)$$

- $\boldsymbol{p}_{\text{col}} = P_{\text{col}}\cdot\boldsymbol{b} \in \mathcal{C}(A)$ is the orthogonal projection of $\boldsymbol{b} \in \mathbb{R}^m$ onto $\mathcal{C}(A)$
- $\boldsymbol{p}_{\text{row}} = \boldsymbol{c}\cdot P_{\text{row}} \in \mathcal{C}(A^\top)$ is the orthogonal projection of $\boldsymbol{c} \in \mathbb{R}^n$ onto $\mathcal{C}(A^\top)$

Orthogonal projection

Recall that, given $A \in \mathbb{R}^{m\times n}$

If $\operatorname{rank}(A) = n$

$$\underset{m\times m}{P^\perp_{\text{col}}} = I_m - \underset{m\times m}{P_{\text{col}}} = \text{projection matrix onto the } \textbf{left nullspace} \text{ of } A, \text{ i.e. } \mathcal{N}(A^\top)$$

If $\operatorname{rank}(A) = m$

$$\underset{n\times n}{P^\perp_{\text{row}}} \coloneqq I_n - \underset{n\times n}{P_{\text{row}}} = \text{projection matrix onto the } \textbf{nullspace} \text{ of } A, \text{ i.e. } \mathcal{N}(A)$$

- $\boldsymbol{p}^\perp_{\text{col}} = P^\perp_{\text{col}}\cdot\boldsymbol{b} \in \mathcal{N}(A^\top)$ is orthogonal to $\boldsymbol{p}_{\text{col}} = P_{\text{col}}\cdot\boldsymbol{b} \in \mathcal{C}(A)$, with $\boldsymbol{b} \in \mathbb{R}^m$
- $\boldsymbol{p}^\perp_{\text{row}} = \boldsymbol{c}\cdot P^\perp_{\text{row}} \in \mathcal{N}(A)$ is orthogonal to $\boldsymbol{p}_{\text{row}} = \boldsymbol{c}\cdot P_{\text{row}} \in \mathcal{C}(A^\top)$

Removing the input terms

The matrix $\mathbf{U}^{\perp}$ is the **orthogonal projection** over the nullspace $\mathcal{N}(\mathbf{U})$, and so is orthogonal to the rowspace $\mathcal{C}(\mathbf{U}^{\top})$

$$\underset{m_u s \times h}{\mathbf{U}} \cdot \underset{h \times h}{\mathbf{U}^{\perp}} = \mathbf{U} \cdot (I_h - \mathbf{U}^{\top}(\mathbf{U}\mathbf{U}^{\top})^{-1}\mathbf{U}) \quad = \mathbf{U} - \mathbf{U}\mathbf{U}^{\top}(\mathbf{U}\mathbf{U}^{\top})^{-1}\mathbf{U} \quad = \mathbf{U} - \mathbf{U} = \underset{m_u s \times h}{\mathbf{0}}$$

So that

$$\underset{ps \times h}{\mathbf{Y}} \cdot \underset{h \times h}{\mathbf{U}^{\perp}} = (\underset{ps \times n}{\mathcal{O}_s} \cdot \underset{n \times h}{\mathbf{X}} + \underset{m_u s \times h}{\overset{ps \times m_u s}{\mathcal{T}_s}} \cdot \underset{h \times h}{\mathbf{U}}) \cdot \underset{h \times h}{\mathbf{U}^{\perp}} \quad \Rightarrow \quad \boxed{\underset{ps \times h}{\mathbf{Y}} \cdot \underset{h \times h}{\mathbf{U}^{\perp}} = \underset{ps \times n}{\mathcal{O}_s} \cdot \underset{n \times h}{\mathbf{X}\mathbf{U}^{\perp}}}$$

This equation shows that $\mathcal{C}(\mathbf{Y}\mathbf{U}^{\perp})$ **is contained into** $\mathcal{C}(\mathcal{O}_s)$. In fact, $\mathcal{O}_s \cdot \mathbf{X}\mathbf{U}^{\perp}$ might have a lower rank than $\mathcal{O}_s$. The next step is to show when $\mathcal{C}(\mathbf{Y}\mathbf{U}^{\perp}) = \mathcal{C}(\mathcal{O}_s)$

Estimation of (A, C)

The following lemma (a sort of **persistency excitation condition**) guarantees that $\mathcal{C}(\mathbf{Y}\mathbf{U}^{\perp}) = \mathcal{C}(\mathcal{O}_s \cdot \mathbf{X}\mathbf{U}^{\perp})$. This is equivalent to show that $\text{rank}(\mathbf{Y}\mathbf{U}^{\perp}) = n$

Lemma

Given a minimal LTI state-space system, if the input $u(\cdot)$ is such that

$$\text{rank}\left(\begin{bmatrix} \overset{n \times (N-s+1)}{\mathbf{X}} \\ \underset{m_u s \times (N-s+1)}{\mathbf{U}} \end{bmatrix}\right) = n + m_u s$$

Then

$$\text{rank}(\mathbf{Y}\mathbf{U}^{\perp}) = n$$

and

$$\mathcal{C}(\mathbf{Y}\mathbf{U}^{\perp}) = \mathcal{C}(\mathcal{O}_s)$$

- Also, in general $n < s \ll N$
- Because this matrix has **full row rank**, adding columns (i.e. data) to it will not change its rank

Estimation of (A, C)

The result

$$\boxed{\mathcal{C}(\mathbf{Y}\mathbf{U}^{\perp}) = \mathcal{C}(\mathcal{O}_s)}$$

implies that $\mathcal{C}(\mathcal{O}_s)$ can be determined by looking for the set of all linear combinations of the columns of matrix $\mathbf{Y}\mathbf{U}^{\perp}$

The elements of $\mathbf{Y}\mathbf{U}^{\perp}$ can **all be evaluated from the data!!**

An **SVD** can be used to determine $\mathcal{C}(\mathbf{Y}\mathbf{U}^{\perp})$, and so (A, C) from $\mathcal{C}(\mathcal{O}_s)$, up to a **similarity matrix** transformation T of the states so that $\boldsymbol{x}_T(t) = T\boldsymbol{x}(t)$

Estimation of (A, C)

Applying the SVD to $\mathbf{Y}\mathbf{U}^{\perp}$, and obtaining a **rank-n approximation**, we get

$$\underset{ps \times (N-s+1)}{\mathbf{Y}\mathbf{U}^{\perp}} = \underset{ps \times n}{\mathcal{O}_s} \cdot \underset{n \times (N-s+1)}{\mathbf{X}\mathbf{U}^{\perp}} \approx \underset{ps \times n}{U_n} \cdot \underset{n \times n}{\Sigma_n} \cdot \underset{n \times (N-s+1)}{V_n^{\top}}$$

where $\text{rank}(\Sigma_n) = \text{rank}(\mathcal{O}_s) = n$. Then, the columns of U_n **spans** $\mathcal{C}(\mathbf{Y}\mathbf{U}^{\perp})$ and also $\mathcal{C}(\mathcal{O}_s)$, so that, up to a **similarity transformation** T where $\boldsymbol{x}_T(t) = T\boldsymbol{x}(t)$

$$U_n = \mathcal{O}_s T^{-1} = \begin{bmatrix} CT^{-1} \\ CT^{-1}(TAT^{-1}) \\ \vdots \\ CT^{-1}(TAT^{-1})^{s-1} \end{bmatrix} = \begin{bmatrix} C_T \\ C_T A_T \\ \vdots \\ C_T A_T^{s-1} \end{bmatrix}$$

Estimation of (A, C)

Hence, similar to the MIMO Kung's algorithm case,

$$\underset{p\times n}{\hat{C}} = \underset{ps\times n}{C_T} = U_n(1{:}\,p, :)$$

$$\underset{n\times n}{\hat{A}} = \underset{n\times n}{A_T} = U_n(1{:}\,(s-1)p, :)^{\dagger} \cdot U_n(p+1{:}\,ps, :)$$

$$n\times(s-1)p \qquad (s-1)p\times n$$

Remark:

A set of system matrices $(\hat{A}, \hat{C})$ with a **different similarity transformation** T can be obtained if we take for example $U_n\Sigma_n^{1/2}$ as the extended observability matrix $\mathcal{O}_s$ of the similarly equivalent state-space model

Outline

1. MIMO subspace identification with generic noiseless I\O data: MOESP
2. Estimating (A, C)
3. **Improving the computational efficiency with RQ factorization**
4. Estimating $(B, D, x(0))$ given $(\hat{A}, \hat{C})$

RQ decomposition

Computing the SVD of

$$\underset{ps\times(N-s+1)}{\mathbf{Y}\mathbf{U}^{\perp}} = \underset{ps\times n}{\mathcal{O}_s} \cdot \underset{n\times(N-s+1)}{\mathbf{X}\mathbf{U}^{\perp}} \approx \underset{ps\times n}{U_n} \cdot \underset{n\times n}{\Sigma_n} \cdot \underset{n\times(N-s+1)}{V_n^{\top}}$$

is **not computationally efficient** because the number of columns of $\mathbf{Y}\mathbf{U}^{\perp}$ is $N-s+1$ and N is typically large

Furthermore, the above equation need the construction of $\mathbf{U}^{\perp}$ (that has $N-s+1$ columns) and $\mathbf{U}^{\perp} = I_h - \mathbf{U}^{\top}(\mathbf{U}\mathbf{U}^{\top})^{-1}\mathbf{U}$ requires the **inversion** of $(\mathbf{U}\mathbf{U}^{\top})$

RQ decomposition

Remark (RQ decomposition from QR decomposition):

Consider a generic matrix $A \in \mathbb{R}^{m\times n}$. Consider the following decompositions

QR decomposition of A: $\underset{m\times n}{A} = \underset{m\times m}{Q_A} \cdot \underset{m\times n}{R_A}$

- R_A is **upper** triangular
- Q_A is orthogonal

RQ decomposition of A: $\underset{m\times n}{A} = \underset{m\times n}{R'_A} \cdot \underset{n\times n}{Q'_A}$ ⇨ $\underset{n\times m}{A^{\top}} = \underset{n\times n}{(Q'_A)^{\top}} \cdot \underset{n\times m}{(R'_A)^{\top}}$

- R'_A is **lower** triangular
- Q'_A is orthogonal

QR decomposition of $A^{\top}$: $\underset{n\times m}{A^{\top}} = \underset{n\times n}{Q_{A^{\top}}} \cdot \underset{n\times m}{R_{A^{\top}}}$

RQ decomposition

Then

$$A^\top = (Q'_A)^\top \cdot (R'_A)^\top = Q_{A^\top} \cdot R_{A^\top}$$

$$\begin{cases} Q'_A = (Q_{A^\top})^\top \\ R'_A = (R_{A^\top})^\top \end{cases}$$

So that the **RQ decomposition** of A can be computed from the **QR decomposition** of $A^\top$

RQ decomposition

For a more efficient implementation with respect both to the **number of flops** and to the required **memory storage**, the **explicit calculation** of the product $\mathbf{YU}^\perp$ can be **avoided** when using the following **RQ factorization:**

$$\underset{ps \times h}{\begin{bmatrix} \mathbf{U} \\ \mathbf{Y} \end{bmatrix}} = \underbrace{\begin{bmatrix} R_{11} & 0 \\ R_{21} & R_{22} \end{bmatrix}}_{R_1} \underbrace{\begin{bmatrix} 0 \\ 0 \end{bmatrix}}_{ps \times ps} \underbrace{\begin{bmatrix} Q_{11} \\ Q_{21} \\ Q_2 \end{bmatrix}}_{Q_1} = R \cdot Q = R_1 \cdot Q_1$$

- $h := N - s + 1$

where $R_{11} \in \mathbb{R}^{m_u s \times m_u s}$ **and** $R_{22} \in \mathbb{R}^{ps \times ps}$. The matrix Q is **orthogonal**, which implies that

$$QQ^\top = \begin{bmatrix} Q_{11} \\ Q_{21} \\ Q_2 \end{bmatrix} \begin{bmatrix} Q_{11}^\top & Q_{21}^\top & Q_2{}^\top \end{bmatrix} = \begin{bmatrix} Q_{11}Q_{11}^\top & Q_{11}Q_{21}^\top & Q_{11}Q_2^\top \\ Q_{21}Q_{11}^\top & Q_{21}Q_{21}^\top & Q_{21}Q_2^\top \\ Q_2Q_{11}^\top & Q_2Q_{21}^\top & Q_2Q_2^\top \end{bmatrix} = I_h$$

RQ decomposition

The previous relationships can be rewritten as

$$\underset{m_u s \times h}{\mathbf{U}} = R_{11}Q_{11}$$

$$\underset{p_u s \times h}{\mathbf{Y}} = \mathcal{O}_s\mathbf{X} + \mathcal{T}_s\mathbf{U} = R_{21}Q_{11} + R_{22}Q_{21}$$

⇨ **The projection matrix $\mathbf{U}^\perp$ can be easily seen to coincide with $Q_{21}^\top$**, since $\mathbf{U} \cdot \mathbf{U}^\perp = R_{11}Q_{11} \cdot Q_{21}^\top = \mathbf{0}$

Furthermore, the last expression can be re-elaborated as follows:

$$\mathcal{O}_s\mathbf{X} + \mathcal{T}_s\mathbf{U} = \mathcal{O}_s\mathbf{X} + \mathcal{T}_s R_{11}Q_{11} = R_{21}Q_{11} + R_{22}Q_{21}$$

and right-multiplying by $Q_{21}^\top$ one gets: $\mathcal{O}_s \cdot \mathbf{X}Q_{21}^\top = R_{22}$

In other words, the **submatrix** R_{22}, which is computed **entirely from the input-output data,** contains information on $\mathcal{O}_s$

RQ decomposition

Under the hypothesis that

$$\underset{m_u s \times (N-s+1)}{\operatorname{rank}\left(\begin{bmatrix} \mathbf{X} \\ \mathbf{U} \end{bmatrix}\right)} = n + m_u s$$

It can be proven that

$$\underset{ps \times ps}{\mathcal{C}(R_{22})} = \underset{ps \times n}{\mathcal{C}(\mathcal{O}_s)}$$

Thus, the SVD of R_{22} can be used to obtain an estimate of $U_n = \mathcal{O}_s T^{-1}$, and thus of matrices (A, C). Note that $R_{22} \in \mathbb{R}^{ps \times ps}$ is smaller than $\mathbf{YU}^\perp \in \mathbb{R}^{ps \times (N-s+1)}$, where typically $N \gg ps$

$$\underset{ps \times ps}{R_{22}} \approx \underset{ps \times n}{U_n} \cdot \overset{n \times n}{\Sigma_n} \cdot \underset{n \times ps}{V_n^\top}$$

- $\operatorname{rank}(\Sigma_n) = n$

Outline

1. MIMO subspace identification with generic noiseless I\O data: MOESP
2. Estimating (A, C)
3. Improving the computational efficiency with RQ factorization
4. **Estimating $(B, D, \boldsymbol{x}(0))$ given $(\hat{A}, \hat{C})$**

Kronecker product

Let $F \in \mathbb{R}^{m \times n}$ and $J \in \mathbb{R}^{r \times s}$ be two rectangular matrices. The **kronecker product** of F and J is an $mr \times ns$ matrix defined as follows:

$$\underset{mr \times ns}{F \otimes J} = \underset{m \times n \quad r \times s}{\begin{bmatrix} f_{11}J & f_{12}J & \cdots & f_{1n}J \\ f_{21}J & f_{22}J & \cdots & f_{2n}J \\ \vdots & \vdots & & \vdots \\ f_{m1}J & f_{m2}J & \cdots & f_{mn}J \end{bmatrix}}$$

- f_{ij} is the (i,j)-th element of F

For any matrices $F \in \mathbb{R}^{m \times n}$, $X \in \mathbb{R}^{n \times s}$, $H \in \mathbb{R}^{s \times p}$, it holds the **«vectorization trick»**

$$\underset{pm \times 1}{\mathrm{vec}(FXH)} = \underset{pm \times sn}{\underset{p \times s}{(H^\top} \otimes \underset{m \times n}{F)} \underset{sn \times 1}{\mathrm{vec}(X)}}$$

where $\mathrm{vec}(F)$ **stacks the columns** of a $m \times n$ matrix F into a $mn \times 1$ **column vector**

Estimation of $(B, D, \boldsymbol{x}(0))$

The matrices $\hat{B} = B_T \coloneqq TB$ and $\hat{D} = D_T \coloneqq D$, along with the initial condition $\hat{\boldsymbol{x}}(0)$, can be computed by solving a **least squares problem**

The output $\boldsymbol{y}(t)$ of the system

$$\mathcal{S}: \begin{cases} \underset{n \times 1}{\boldsymbol{x}_T(t+1)} = \underset{n \times n}{A_T}\boldsymbol{x}(t) + \underset{n \times m_u}{B_T}\underset{m_u \times 1}{\boldsymbol{u}(t)} \\ \underset{p \times 1}{\boldsymbol{y}(t)} = \underset{p \times n}{C_T}\boldsymbol{x}(t) + \underset{p \times m_u}{D_T}\underset{m_u \times 1}{\boldsymbol{u}(t)} \end{cases} \qquad \boldsymbol{x}_T(0)$$

can be computed as

$$\underset{p \times 1}{\boldsymbol{y}(t)} = \sum_{\tau=0}^{t-1} \underset{p \times n}{\overset{n \times n}{C_T A_T^{t-\tau-1}}} \underset{n \times m_u}{B_T} \cdot \underset{m_u \times 1}{\boldsymbol{u}(\tau)} + \underset{p \times m_u}{D_T}\boldsymbol{u}(t) + \underset{n \times 1}{C_T A_T^t \boldsymbol{x}_T(0)}$$

Estimation of $(B, D, \boldsymbol{x}(0))$

Using the Kronecker product on the outputs expression:

$$\mathrm{vec}(FXH) = (H^\top \otimes F)\mathrm{vec}(X) \qquad \boldsymbol{y}(t) = \sum_{\tau=0}^{t-1} C_T A_T^{t-\tau-1} B_T \cdot \boldsymbol{u}(\tau) + D_T \boldsymbol{u}(t) + C_T A_T^t \boldsymbol{x}_T(0)$$

$$\boldsymbol{y}(t) = \left(\sum_{\tau=0}^{t-1} \boldsymbol{u}(\tau)^\top \otimes C_T A_T^{t-\tau-1} \right) \mathrm{vec}(B_T) + \left(\boldsymbol{u}(t)^\top \otimes I_p\right)\mathrm{vec}(D_T) + C_T A_T^t \boldsymbol{x}_T(0)$$

$$\underset{p \times 1}{\boldsymbol{y}(t)} = \underset{p \times (nm_u + pm_u + n)}{\Phi^\top(t)} \cdot \underset{(nm_u + pm_u + n) \times 1}{\boldsymbol{\theta}} = \begin{bmatrix} \underset{p \times m_u n}{\sum_{\tau=0}^{t-1} \underset{1 \times m_u}{\boldsymbol{u}(\tau)^\top} \otimes \underset{p \times n}{C_T A_T^{t-\tau-1}}} & \underset{p \times m_u p}{\underset{1 \times m_u}{\boldsymbol{u}(t)^\top} \otimes \underset{p \times p}{I_p}} & \underset{p \times n}{C_T A_T^t} \end{bmatrix} \begin{bmatrix} \underset{nm_u \times 1}{\mathrm{vec}(B_T)} \\ \underset{pm_u \times 1}{\mathrm{vec}(D_T)} \\ \underset{n \times 1}{\boldsymbol{x}_T(0)} \end{bmatrix}$$

Estimation of $(B, D, \boldsymbol{x}(0))$

The estimation of $\boldsymbol{\theta}$ can be computed in a **(multi-output) least squares** setting

$$\widehat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \sum_{t=0}^{N-1} \|\boldsymbol{y}(t) - \Phi^{\mathsf{T}}(t)\boldsymbol{\theta}\|_2^2$$

And the solution is

$$\widehat{\boldsymbol{\theta}} = \left(\sum_{t=0}^{N-1} \Phi(t) \cdot \Phi^{\mathsf{T}}(t)\right)^{-1} \cdot \sum_{t=0}^{N-1} \Phi(t) \cdot \boldsymbol{y}(t)$$

Example: simple SISO MOESP deterministic case

Consider the **deterministic** system $\mathcal{S}: \begin{cases} x(t+1) = 0.1x(t) + 0.1u(t) \\ y(t) = x(t) \end{cases} \quad x(0) = 0$

and assume that the system is subject to the following input samples

$[u(0) \quad u(1) \quad u(2) \quad u(3)]^{\mathsf{T}} = [1 \quad 2 \quad 1 \quad 1]^{\mathsf{T}}$

starting from **null initial conditions**, thus resulting in the corresponding outputs:

$[y(0) \quad y(1) \quad y(2) \quad y(3)]^{\mathsf{T}} = [0 \quad 0.1 \quad 0.21 \quad 0.121]^{\mathsf{T}}$

Then, we can construct $\mathbf{U}$ and $\mathbf{Y}$ as follows ($s = 2$):

$$\mathbf{U} = \begin{bmatrix} u(0) & u(1) & u(2) \\ u(1) & u(2) & u(3) \end{bmatrix} = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 1 & 1 \end{bmatrix} \qquad \mathbf{Y} = \begin{bmatrix} y(0) & y(1) & y(2) \\ y(1) & y(2) & y(3) \end{bmatrix} = \begin{bmatrix} 0 & 0.1 & 0.21 \\ 0.1 & 0.21 & 0.121 \end{bmatrix}$$

$m_u s \times (N - s + 1) = m_u s \times h$
$2 \times (4 - 2 + 1) = 2 \times 3$

$ps \times (N - s + 1) = ps \times h$
$2 \times (4 - 2 + 1) = 2 \times 3$

Example: simple SISO MOESP deterministic case

$$\underset{3\times 3}{\mathbf{U}^{\perp}} = \underset{(N-s+1)\times(N-s+1)}{I_3} - \underset{(N-s+1)\times s}{\mathbf{U}^{\mathsf{T}}} \underset{3\times 2}{(\mathbf{U}\mathbf{U}^{\mathsf{T}})^{-1}} \underset{2\times 3}{\mathbf{U}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} - \begin{bmatrix} 1 & 2 \\ 2 & 1 \\ 1 & 1 \end{bmatrix} \left(\begin{bmatrix} 1 & 2 & 1 \\ 2 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 2 & 1 \\ 1 & 1 \end{bmatrix} \right)^{-1} \cdot \begin{bmatrix} 1 & 2 & 1 \\ 2 & 1 & 1 \end{bmatrix} =$$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} - \begin{bmatrix} 1 & 2 \\ 2 & 1 \\ 1 & 1 \end{bmatrix} \left(\begin{bmatrix} 6 & 5 \\ 5 & 6 \end{bmatrix} \right)^{-1} \cdot \begin{bmatrix} 1 & 2 & 1 \\ 2 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} - \frac{1}{11} \begin{bmatrix} 1 & 2 \\ 2 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 6 & -5 \\ -5 & 6 \end{bmatrix} \cdot \begin{bmatrix} 1 & 2 & 1 \\ 2 & 1 & 1 \end{bmatrix} =$$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} - \frac{1}{11} \begin{bmatrix} 1 & 2 \\ 2 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} -4 & 7 & 1 \\ 7 & -4 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} - \frac{1}{11} \begin{bmatrix} 10 & -1 & 3 \\ -1 & 10 & 3 \\ 3 & 3 & 2 \end{bmatrix} =$$

$$= \frac{1}{11} \begin{bmatrix} 1 & 1 & -3 \\ 1 & 1 & -3 \\ -3 & -3 & 9 \end{bmatrix}$$

Example: simple SISO MOESP deterministic case

Verify that $\underset{2\times 3}{\mathbf{U}^{\perp}}$ is such that $\underset{3\times 2}{\mathbf{U}} \underset{3\times 3}{\mathbf{U}^{\perp}} = \mathbf{0}$ ⇨ $\mathbf{U}\mathbf{U}^{\perp} = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 1 & 1 \end{bmatrix} \cdot \frac{1}{11} \begin{bmatrix} 1 & 1 & -3 \\ 1 & 1 & -3 \\ -3 & -3 & 9 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$

Now,

$$\underset{2\times 3}{\mathbf{Y}} \cdot \underset{3\times 3}{\mathbf{U}^{\perp}} = \begin{bmatrix} 0 & 0.1 & 0.21 \\ 0.1 & 0.21 & 0.121 \end{bmatrix} \cdot \frac{1}{11} \begin{bmatrix} 1 & 1 & -3 \\ 1 & 1 & -3 \\ -3 & -3 & 9 \end{bmatrix} = \frac{1}{11} \begin{bmatrix} -0.53 & -0.53 & 1.59 \\ -0.053 & -0.053 & 0.159 \end{bmatrix}$$

There is no need to apply the SVD to recognize that $\mathbf{Y}\mathbf{U}^{\perp}$ has rank 1 (the 2nd row is one tenth of the 1st). Still, let's do the SVD

$$\mathbf{Y}\mathbf{U}^{\perp} = \underset{\substack{U \\ ps\times s \\ 2\times 2}}{\begin{bmatrix} 0.9950 & -0.0995 \\ 0.0995 & 0.9950 \end{bmatrix}} \underset{\substack{\Sigma \\ s\times h \\ 2\times 3}}{\begin{bmatrix} 0.1606 & 0 & 0 \\ 0 & 1 \cdot 10^{-16} & 0 \end{bmatrix}} \underset{\substack{V^{\mathsf{T}} \\ h\times h \\ 3\times 3}}{\begin{bmatrix} -0.3015 & -0.3015 & 0.9045 \\ 0.2958 & -0.9314 & -0.2119 \\ 0.9064 & 0.2037 & 0.3700 \end{bmatrix}}$$

Example: simple SISO MOESP deterministic case

Since $n = 1$, we can set $\hat{O}_s = U_n = U(:,1)$ (but we can also select the 1° column of $\mathbf{YU}^{\perp}$)

$$U_1 = \begin{bmatrix} 0.9950 \\ 0.0995 \end{bmatrix} \Rightarrow$$

$$\hat{C} = U_n(1,:) = 0.9950 \approx 1$$

$$\hat{A} = U_n(1:(s-1)p,:)^{\dagger} \cdot U_n(p+1:sp,:) = \frac{-0.0995}{-0.9950} = 0.1$$

Example: simple SISO MOESP deterministic case

To find $\hat{B}$ and $\boldsymbol{x}(0)$, we construct a **linear regression**

$$\boldsymbol{y}(t) = \Phi^{\top}(t) \cdot \boldsymbol{\theta} = \underbrace{\left[\sum_{\tau=0}^{t-1} \underset{1\times m_u}{\boldsymbol{u}(\tau)^{\top}} \otimes \underset{p\times n}{C_T A_T^{t-\tau-1}}\right.}_{\Phi_1^{\top}(t)} \underbrace{\left.\underset{1\times m_u}{\boldsymbol{u}(t)^{\top}} \otimes \underset{p\times p}{I}\right.}_{\Phi_2^{\top}(t)} \underbrace{\left.\underset{p\times n}{C_T A_T^{t}}\right]}_{\Phi_3^{\top}(t)} \begin{bmatrix} \mathrm{vec}(B_T) \\ \mathrm{vec}(D_T) \\ \boldsymbol{x}'(0) \end{bmatrix}$$

$\underset{p\times m_u n}{}$ $\underset{p\times m_u p}{}$

$$\boldsymbol{\Phi}_1^{\top} = [0 \quad \hat{C}u(0) \quad \hat{C}\hat{A}u(0) + \hat{C}u(1) \quad \hat{C}\hat{A}^2u(0) + \hat{C}\hat{A}u(1) + \hat{C}u(2)]$$
$$= [0 \quad 0.9950 \quad 2.0896 \quad 1.2040]$$

$\boldsymbol{\Phi}_1^{\top}$ is built from stacking the $\Phi_1^{\top}(t)$ for several times t

$\boldsymbol{\Phi}_2^{\top}$ = not present

$$\boldsymbol{\Phi}_3^{\top} = [\hat{C} \quad \hat{C}\hat{A} \quad \hat{C}\hat{A}^2 \quad \hat{C}\hat{A}^3]$$
$$= [0.995 \quad 0.0995 \quad 0.009950 \quad 0.0009950]$$

Example: simple SISO MOESP deterministic case

Solving the Least Squares problem:

$$\hat{\boldsymbol{\theta}} = (\boldsymbol{\Phi}^{\top}\boldsymbol{\Phi})^{-1} \cdot \boldsymbol{\Phi}^{\top}\boldsymbol{y} = \left(\begin{bmatrix} 0 & 0.9950 & 2.0896 & 1.2040 \\ 0.995 & 0.0995 & 0.00995 & 0.000995 \end{bmatrix} \begin{bmatrix} 0 & 0.995 \\ 0.9950 & 0.0995 \\ 2.0896 & 0.00995 \\ 1.2040 & 0.000995 \end{bmatrix} \right)^{-1}$$
$$\cdot \begin{bmatrix} 0 & 0.9950 & 2.0896 & 1.2040 \\ 0.995 & 0.0995 & 0.00995 & 0.000995 \end{bmatrix} \begin{bmatrix} 0 \\ 0.1 \\ 0.21 \\ 0.121 \end{bmatrix} = \begin{bmatrix} 0.1005 \\ 0 \end{bmatrix}$$

leads to $\hat{B} \approx 0.1$ and $\hat{x}(0) = 0$

Notice that $G(z) = \hat{C}(zI - \hat{A})^{-1}B = \frac{0.1}{z-0.1}$, which **equals the true transfer function**

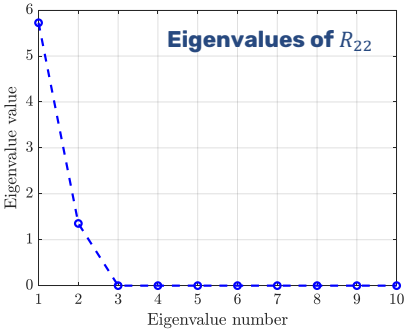
Example: MIMO MOESP deterministic case

Consider the system:

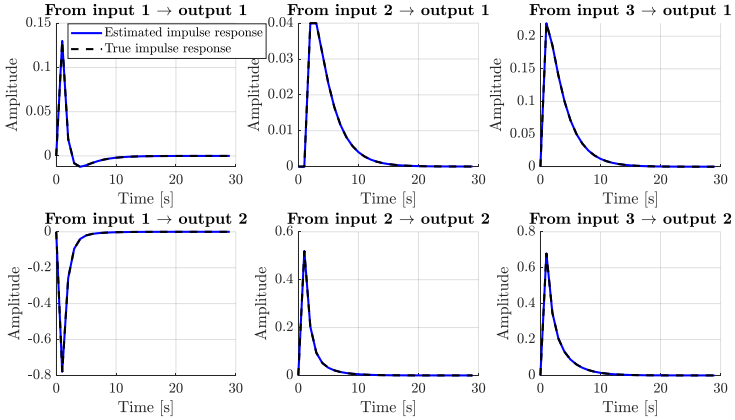
$$\mathcal{S}: \begin{cases} \boldsymbol{x}(t+1) = A\boldsymbol{x}(t) + B\boldsymbol{u}(t) \\ \boldsymbol{y}(t) = C\boldsymbol{x}(t) \end{cases} \quad A = \begin{bmatrix} 0.7 & 0 \\ 0 & 0.3 \end{bmatrix} \quad B = \begin{bmatrix} -0.1 & 0.2 & 0.6 \\ 0.9 & -0.5 & -0.4 \end{bmatrix} \quad C = \begin{bmatrix} 0.5 & 0.2 \\ 0.6 & -0.8 \end{bmatrix}$$

Measure $N = 30$ data using a white noise input $u(t) \sim \mathrm{WN}(0,1)$

Apply the RQ decomposition and then perform SVD on the matrix R_{22}



Example: MIMO MOESP deterministic case



UNIVERSITÀ DEGLI STUDI DI BERGAMO | Dipartimento di Ingegneria Gestionale, dell'Informazione e della Produzione

ADAPTIVE LEARNING, ESTIMATION AND SUPERVISION OF DYNAMICAL SYSTEMS (ALES)

Lecture 8: Multivariable Output Error State Space (MOESP) subspace identification - part II

Master Degree in COMPUTER ENGINEERING

Data Science and Data Engineering Curriculum

SPEAKER
Prof. Mirko Mazzoleni

PLACE
University of Bergamo

Syllabus

1. **Recursive and adaptive identification**
   - 1.1 Recursive ARX estimation (RLS)
   - 1.2 Least Mean Squares (LMS)
   - 1.3 Instrumental Variables (IV)
2. **Closed-loop identification**
3. **Subspace and MIMO identification**
   - 3.1 Singular Value Decomposition
   - 3.2 Impulse data: Ho-Kalman, Kung algorithms
   - 3.3 Generic I/O data: the MOESP algorithm
4. **Supervision of dynamical systems**
   - 4.1 Introduction to fault diagnosis
   - 4.2 Model-based fault diagnosis
   - 4.3 Parity space approaches
   - 4.4 Observer-based approaches
   - 4.5 Signal-based fault diagnosis
   - 4.6 Knowledge-based fault diagnosis

CASE STUDIES

- Virtual Reference Feedback Tuning
- Nuclear particles classification
- Leak detection in an industrial valve
- Bearing fault identification

Outline

1. MIMO subspace identification for stochastic systems
2. Output-error case: output white-noise
3. General case: process and output white-noises

Outline

1. **MIMO subspace identification for stochastic systems**
2. Output-error case: output white-noise
3. General case: process and output white-noises

MOESP algorithm for stochastic systems

Consider the **stochastic** MIMO system

$$\mathcal{S}:\begin{cases}\underset{n\times 1}{\boldsymbol{x}(t+1)} = \underset{n\times n}{A}\boldsymbol{x}(t) + \underset{n\times m_u}{B}\underset{m_u\times 1}{\boldsymbol{u}(t)}\\ \underset{p\times 1}{\boldsymbol{y}(t)} = \underset{p\times n}{C}\boldsymbol{x}(t) + \underset{p\times m_u}{D}\underset{m_u\times 1}{\boldsymbol{u}(t)} + \underset{p\times 1}{\boldsymbol{v}_2(t)}\end{cases} \qquad \boldsymbol{x}(0); \qquad \boldsymbol{v}_2(t)\perp\boldsymbol{u}(t)$$

where $G(z) = C(zI_n - A)^{-1}B + D$ and $\boldsymbol{v}_2(t)$ represents the output **stationary** (and ergodic) noise disturbance, which may also be a **colored** noise

An equivalent formulation is the so-called **innovation form,** with K_0 the **steady-state Kalman gain**

$$\mathcal{S}:\begin{cases}\boldsymbol{\xi}(t+1) = A\boldsymbol{\xi}(t) + B\boldsymbol{u}(t) + \underset{m_u\times p}{K_0}\boldsymbol{e}(t)\\ \boldsymbol{y}(t) = C\boldsymbol{\xi}(t) + D\boldsymbol{u}(t) + \underset{p\times 1}{\boldsymbol{e}(t)}\end{cases} \qquad \boldsymbol{\xi}(0); \qquad \begin{matrix}\boldsymbol{e}(t)\sim \mathrm{WN}(\boldsymbol{0}, \underset{p\times p}{V_2})\\ \boldsymbol{e}(t)\perp\boldsymbol{u}(t)\end{matrix}$$

MOESP algorithm for stochastic systems

A further, most general, alternative formulation is

$$\mathcal{S}:\begin{cases}\boldsymbol{x}(t+1) = A\boldsymbol{x}(t) + B\boldsymbol{u}(t) + \boldsymbol{v}_1(t)\\ \boldsymbol{y}(t) = C\boldsymbol{x}(t) + D\boldsymbol{u}(t) + \boldsymbol{v}_2(t)\end{cases} \qquad \boldsymbol{x}(0); \qquad \begin{matrix}\boldsymbol{v}_1(t)\sim \mathrm{WN}(\boldsymbol{0}, V_1)\\ \boldsymbol{v}_2(t)\sim \mathrm{WN}(\boldsymbol{0}, V_2)\\ \boldsymbol{v}_1, \boldsymbol{v}_2(t)\perp\boldsymbol{u}(t)\end{matrix} \qquad \mathbb{E}\left[\begin{bmatrix}\boldsymbol{v}_1\\ \boldsymbol{v}_2\end{bmatrix}\begin{bmatrix}\boldsymbol{v}_1 & \boldsymbol{v}_2\end{bmatrix}\right] = \begin{bmatrix}V_1 & V_{12}\\ V_{12}^{\mathrm{T}} & V_2\end{bmatrix}$$

We will consider two cases:

1. **White output** noise (output-error model) with **independent components,**

$$\boldsymbol{v}_1(t) = \boldsymbol{0}, \qquad \boldsymbol{v}_2(t)\sim \mathrm{WN}\left(\boldsymbol{0}, \lambda^2 I_p\right)$$

2. **White process** and **white output** noise, $\boldsymbol{v}_1(t)\sim \mathrm{WN}(\boldsymbol{0}, V_1)$, $\boldsymbol{v}_2(t)\sim \mathrm{WN}(\boldsymbol{0}, V_2)$

Outline

1. MIMO subspace identification for stochastic systems

2. **Output-error case: output white-noise**

3. General case: process and output white-noises

MOESP algorithm for output white noise error

First, consider the **output-error white noise case**. The system to be considered is

$$\mathcal{S}:\begin{cases} \underset{n\times 1}{\boldsymbol{x}(t+1)} = \underset{n\times n}{A}\boldsymbol{x}(t) + \underset{n\times m_u}{B}\underset{m_u\times 1}{\boldsymbol{u}(t)} \\ \underset{p\times 1}{\boldsymbol{y}(t)} = \underset{p\times n}{C}\boldsymbol{x}(t) + \underset{p\times m_u}{D}\underset{m_u\times 1}{\boldsymbol{u}(t)} + \underset{p\times 1}{\boldsymbol{v}_2(t)} \end{cases} \quad \boldsymbol{x}(0); \qquad \begin{aligned} &\boldsymbol{v}_2(t) \sim \mathrm{WN}\left(\mathbf{0}, \underset{p\times p}{\lambda^2 I_p}\right) \\ &\boldsymbol{v}_2(t) \perp \boldsymbol{u}(t) \end{aligned}$$

Stack the noise samples over a time window of length s as

$$\underset{ps\times 1}{\overrightarrow{\boldsymbol{v}_{2,s}}(t)} = \begin{bmatrix} \boldsymbol{v}_2(t) \\ \boldsymbol{v}_2(t+1) \\ \boldsymbol{v}_2(t+2) \\ \vdots \\ \boldsymbol{v}_2(t+s-1) \end{bmatrix} \quad \Rightarrow \quad \underset{ps\times(N-s+1)}{\mathbf{V}} \coloneqq \left[\overrightarrow{\boldsymbol{v}_{2,s}}(1) \quad \overrightarrow{\boldsymbol{v}_{2,s}}(2) \quad \cdots \quad \overrightarrow{\boldsymbol{v}_{2,s}}(N-s+1)\right]$$

MOESP algorithm for output white noise error

The data equation for this system reads as

$$\underset{ps\times(N-s+1)}{\mathbf{Y}} = \underset{ps\times n}{\mathcal{O}_s} \cdot \overset{n\times(N-s+1)}{\mathbf{X}} + \underset{ps\times m_u s}{\mathcal{T}_s} \cdot \overset{m_u s\times(N-s+1)}{\mathbf{U}} + \underset{ps\times(N-s+1)}{\mathbf{V}}$$

The following Lemma shows that, if the limit $N \to +\infty$, the previous MOESP **results without noise still hold** in this **output-error white noise** case

MOESP algorithm for output white noise error

Lemma

Given a minimal LTI state-space system with white output noise $\boldsymbol{v}_2(t) \sim \mathrm{WN}\left(0, \lambda^2 I_p\right)$ so that $v_2(t) \perp u(t)$, if the input $u(\cdot)$ is such that

$$\operatorname{rank}\left(\lim_{N\to+\infty}\left(\frac{1}{N}\underset{(n+m_u s)\times(n+m_u s)}{\begin{bmatrix}\mathbf{X}\\ \mathbf{U}\end{bmatrix}}\begin{bmatrix}\mathbf{X}^\top & \mathbf{U}^\top\end{bmatrix}\right)\right) = n + m_u s$$

Then, the SVD of $\lim_{N\to+\infty}\left(\frac{1}{N}\mathbf{Y}\mathbf{U}^\perp\mathbf{Y}^\top\right)$ is

$$\lim_{N\to+\infty}\left(\frac{1}{N}\underset{ps\times ps}{\mathbf{Y}\mathbf{U}^\perp\mathbf{Y}^\top}\right) = \begin{bmatrix}U_1 & U_2\end{bmatrix}\begin{bmatrix}\Sigma_n + \lambda^2 I_n & 0 \\ 0 & \lambda^2 I_{ps-n}\end{bmatrix}\begin{bmatrix}U_1^\top \\ U_2^\top\end{bmatrix}$$

MOESP algorithm for output white noise error

Furtheremore, the matrix U_n has rank n and satisfies

$$\mathcal{C}(U_n) = \mathcal{C}(\mathcal{O}_s)$$

As in the noiseless case, we can compute the RQ factorization of

$$\begin{bmatrix} \mathbf{U} \\ \mathbf{Y} \end{bmatrix} = \begin{bmatrix} R_{11} & 0 & 0 \\ R_{21} & R_{22} & 0 \end{bmatrix} \begin{bmatrix} Q_{11} \\ Q_{21} \\ Q_2 \end{bmatrix}$$

And the SVD of R_{22} to obtain its rank-n approximation

$$R_{22} \approx U_n \cdot \Sigma_n \cdot V_n^\top$$

MOESP algorithm for output white noise error

The SVD of R_{22}, as $N \to +\infty$, looks like:

$$\lim_{N\to+\infty} \frac{1}{\sqrt{N}} \underset{ps \times ps}{R_{22}} = [U_1 \quad U_2] \begin{bmatrix} \sqrt{\Sigma_n + \lambda^2 I_n} & 0 \\ 0 & \sqrt{\lambda^2} I_{ps-n} \end{bmatrix} \begin{bmatrix} U_1^\top \\ U_2^\top \end{bmatrix}$$

so that it also provides an **estimate of the noise variance** λ^2 (or, better speaking, of the noise standard deviation $\sqrt{\lambda^2}$) by looking at the **lowest singular values**

Outline

1. MIMO subspace identification for stochastic systems
2. Output-error case: output white-noise
3. **General case: process and output white-noises**

MOESP algorithm for general stochastic systems

We now consider systems of general form

$$\mathcal{S}: \begin{cases} \boldsymbol{x}(t+1) = A\boldsymbol{x}(t) + B\boldsymbol{u}(t) + \boldsymbol{v}_1(t) \\ \boldsymbol{y}(t) = C\boldsymbol{x}(t) + D\boldsymbol{u}(t) + \boldsymbol{v}_2(t) \end{cases} \quad \boldsymbol{x}(0); \quad \begin{array}{l} \boldsymbol{v}_1(t) \sim \text{WN}(\mathbf{0}, V_1) \\ \boldsymbol{v}_2(t) \sim \text{WN}(\mathbf{0}, V_2) \\ \boldsymbol{v}_1, \boldsymbol{v}_2(t) \perp \boldsymbol{u}(t) \end{array} \quad \mathbb{E}\left[\begin{bmatrix} \boldsymbol{v}_1 \\ \boldsymbol{v}_2 \end{bmatrix} [\boldsymbol{v}_1 \quad \boldsymbol{v}_2]\right] = \begin{bmatrix} V_1 & V_{12} \\ V_{12}^\top & V_2 \end{bmatrix}$$

Again, the data equations is:

$$\underset{ps \times (N-s+1)}{\mathbf{Y}} = \underset{ps \times n}{\mathcal{O}_s} \cdot \overset{n \times (N-s+1)}{\mathbf{X}} + \underset{ps \times m_u s}{\mathcal{T}_s} \cdot \overset{m_u s \times (N-s+1)}{\mathbf{U}} + \underset{ps \times (N-s+1)}{\mathbf{V}}$$

MOESP algorithm for general stochastic systems

Remove the input-dependent term by right-multiplying for $\mathbf{U}^\perp$, so that

$$\underset{ps\times h}{\mathbf{Y}}\cdot\underset{h\times h}{\mathbf{U}^\perp}=\underset{ps\times n}{\mathcal{O}_s}\cdot\underset{n\times h}{\mathbf{X}}\cdot\underset{h\times h}{\mathbf{U}^\perp}+\overset{ps\times m_u s}{\mathcal{T}_s}\cdot\underset{m_u s\times h}{\mathbf{U}}\cdot\underset{h\times h}{\mathbf{U}^\perp}+\underset{ps\times h}{\mathbf{V}}\cdot\underset{h\times h}{\mathbf{U}^\perp}$$

$$=\mathcal{O}_s\cdot\mathbf{X}\cdot\mathbf{U}^\perp+\mathbf{V}\cdot\mathbf{U}^\perp$$

- $h \coloneqq N-s+1$

Now, in order to estimate $\mathcal{O}_s$ from $\mathbf{Y}\mathbf{U}^\perp$, the noise term $\mathbf{V}\mathbf{U}^\perp$ should be removed. The idea is to use an **instrumental variable matrix** $\mathbf{Z}\in\mathbb{R}^{n_z s\times(N-s+1)}$ so that

$$\mathbf{Z}=[\boldsymbol{z}(1)\quad\boldsymbol{z}(2)\quad\cdots\quad\boldsymbol{z}(N-s+1)];\qquad\boldsymbol{z}(i)\in\mathbb{R}^{n_z s\times 1}$$

where the $\boldsymbol{z}$ vectors have still to be defined

MOESP algorithm for general stochastic systems

Right-multiplying for $\mathbf{Z}^\top$ and normalizing for N we get

$$\frac{1}{N}\mathbf{Y}\mathbf{U}^\perp\cdot\mathbf{Z}^\top=\frac{1}{N}\mathcal{O}_s\mathbf{X}\mathbf{U}^\perp\cdot\mathbf{Z}^\top+\frac{1}{N}\mathbf{V}\mathbf{U}^\perp\cdot\mathbf{Z}^\top$$

The instruments matrix $\mathbf{Z}$ should be designed such that

$$\lim_{N\to+\infty}\frac{1}{N}\underset{ps\times h}{\mathbf{V}\mathbf{U}^\perp}\cdot\underset{h\times n_z s}{\mathbf{Z}^\top}=\mathbf{0}\qquad\textbf{and}\qquad\lim_{N\to+\infty}\frac{1}{N}\underset{n\times h}{\mathbf{X}\mathbf{U}^\perp}\cdot\underset{h\times n_z s}{\mathbf{Z}^\top}=\tilde{T}\neq\mathbf{0}$$

where $\tilde{T}$ has **full row rank** n

MOESP algorithm for general stochastic systems

Using the instruments matrix $\mathbf{Z}$ we have

$$\lim_{N\to+\infty}\frac{1}{N}\mathbf{Y}\mathbf{U}^\perp\mathbf{Z}^\top=\lim_{N\to+\infty}\frac{1}{N}\mathcal{O}_s\mathbf{X}\mathbf{U}^\perp\mathbf{Z}^\top$$

and since we assumed $\operatorname{rank}\left(\lim_{N\to+\infty}\frac{1}{N}\mathcal{O}_s\mathbf{X}\mathbf{U}^\perp\mathbf{Z}^\top\right)=n$, it follows that

$$\mathcal{C}\left(\lim_{N\to+\infty}\frac{1}{N}\underset{ps\times n_z s}{\mathbf{Y}\mathbf{U}^\perp\mathbf{Z}^\top}\right)=\underset{ps\times h}{\mathcal{C}(\mathcal{O}_s)}$$

Again, an estimate of $\mathcal{C}(\mathcal{O}_s)$ can be obtained by the SVD of $\mathbf{Y}\mathbf{U}^\perp\mathbf{Z}^\top$

RQ decomposition

For an efficient implementation, we can again use the RQ decomposition, as

$$\begin{matrix}m_u s\times h\\ n_z s\times h\\ ps\times h\end{matrix}\begin{bmatrix}\mathbf{U}\\ \mathbf{Z}\\ \mathbf{Y}\end{bmatrix}=\overbrace{\begin{bmatrix}R_{11} & 0 & 0\\ R_{21} & R_{22} & 0\\ R_{31} & R_{32} & R_{33}\end{bmatrix}}^{R_1}\begin{bmatrix}0\\0\\0\end{bmatrix}\overbrace{\begin{bmatrix}Q_{11}\\Q_{21}\\Q_{31}\\Q_4\end{bmatrix}}^{Q_1}=\underset{(m_u s+n_z s+ps)\times h}{R\cdot Q}=\underset{(m_u s+n_z s+ps)\times(m_u s+n_z s+ps)}{R_1}\cdot\underset{(m_u s+n_z s+ps)\times h}{Q_1}$$

- $h \coloneqq N-s+1$

$$\Rightarrow\quad\begin{cases}\mathbf{U}=R_{11}Q_{11}\\ \mathbf{Z}=R_{21}Q_{11}+R_{22}Q_{21}\\ \mathbf{Y}=R_{31}Q_{11}+R_{32}Q_{21}+R_{33}Q_{31}\end{cases}$$

RQ decomposition

From the definition of Q we have that

$$QQ^\top = \begin{bmatrix} Q_{11} \\ Q_{21} \\ Q_{31} \\ Q_4 \end{bmatrix} [Q_{11}^\top \;\; Q_{21}^\top \;\; Q_{31}^\top \;\; Q_4^\top] = I_h$$

$$Q^\top Q = [Q_{11}^\top \;\; Q_{21}^\top \;\; Q_{31}^\top \;\; Q_{41}^\top] \begin{bmatrix} Q_{11} \\ Q_{21} \\ Q_{31} \\ Q_4 \end{bmatrix} = Q_{11}^\top Q_{11} + Q_{21}^\top Q_{21} + Q_{31}^\top Q_{31} + Q_{41}^\top Q_{41} = I_h$$

RQ decomposition

Then,

$$\mathbf{U}^\perp = I_h - \mathbf{U}^\top(\mathbf{U}\mathbf{U}^\top)^{-1}\mathbf{U} = I_h - Q_{11}^\top R_{11}^\top (R_{11}Q_{11}Q_{11}^\top R_{11}^\top)^{-1} R_{11}Q_{11} = I_h - Q_{11}^\top Q_{11}$$

$$= Q_{21}^\top Q_{21} + Q_{31}^\top Q_{31} + Q_4^\top Q_4$$

So,

$$\mathbf{Y}\mathbf{U}^\perp\mathbf{Z}^\top = (R_{31}Q_{11} + R_{32}Q_{21} + R_{33}Q_{31}) \cdot (Q_{21}^\top Q_{21} + Q_{31}^\top Q_{31} + Q_4^\top Q_4) \cdot (Q_{11}^\top R_{21}^\top + Q_{21}^\top R_{22}^\top)$$

$$= (R_{32}Q_{21} + R_{33}Q_{31}) \cdot (Q_{11}^\top R_{21}^\top + Q_{21}^\top R_{22}^\top)$$

$$= R_{32}R_{22}^\top$$

RQ decomposition

So, if the **properties** of the **instruments matrix** $\mathbf{Z}$ hold, and $\lim_{N\to+\infty} \frac{1}{\sqrt{N}} R_{22}$ is **invertible**, we have that

$$\mathcal{C}\left(\lim_{N\to+\infty} \frac{1}{N} \underset{ps\times h}{\mathbf{Y}\mathbf{U}^\perp\mathbf{Z}^\top}\right) = \mathcal{C}\left(\lim_{N\to+\infty} \frac{1}{\sqrt{N}} \underset{ps\times n_z s}{R_{32}}\right) = \underset{ps\times n}{\mathcal{C}(\mathcal{O}_s)}$$

Hence, the matrix $\lim_{N\to+\infty} \frac{1}{\sqrt{N}} R_{32}$ can be used to obtain asymptotically unbiased estimates of the matrices A_T and C_T. The estimation of B_T and D_T is analogous to the deterministic case

Choice of the instruments

The choice of $\mathbf{Z}$ must satisfy

$$\lim_{N\to+\infty} \frac{1}{N} \mathbf{V}\mathbf{U}^\perp \cdot \mathbf{Z}^\top = \mathbf{0} \qquad \textbf{and} \qquad \lim_{N\to+\infty} \frac{1}{N} \underset{n\times h}{\mathbf{X}\mathbf{U}^\perp} \cdot \mathbf{Z}^\top = \tilde{T} \neq \mathbf{0}$$

where $\tilde{T}$ has **full row rank** n

Let's focus on the expression of the first condition

$$\mathbf{V} \cdot \mathbf{U}^\perp \cdot \mathbf{Z}^\top = \mathbf{V} \cdot (I_h - \mathbf{U}^\top(\mathbf{U}\mathbf{U}^\top)^{-1}\mathbf{U}) \cdot \mathbf{Z}^\top = (\mathbf{V} - \mathbf{V}\mathbf{U}^\top(\mathbf{U}\mathbf{U}^\top)^{-1}\mathbf{U})\mathbf{Z}^\top$$

$$= \mathbf{V}\mathbf{Z}^\top - \mathbf{V}\mathbf{U}^\top(\mathbf{U}\mathbf{U}^\top)^{-1}\mathbf{U}\mathbf{Z}^\top$$

Choice of the instruments

Thus, the first condition becomes

$$\lim_{N\to+\infty}\frac{1}{N}\mathbf{V}\mathbf{U}^{\perp}\cdot\mathbf{Z}^{\top}=\lim_{N\to+\infty}\frac{1}{N}(\mathbf{V}\mathbf{Z}^{\top}-\mathbf{V}\mathbf{U}^{\top}(\mathbf{U}\mathbf{U}^{\top})^{-1}\mathbf{U}\mathbf{Z}^{\top})=\mathbf{0}$$

By assumption, we have that $\boldsymbol{v}_2 \perp \boldsymbol{u}(t)$, which is fine in **open-loop** settings. It follows that the first condition simplifies to

$$\lim_{N\to+\infty}\frac{1}{N}\mathbf{V}\mathbf{U}^{\perp}\cdot\mathbf{Z}^{\top}=\lim_{N\to+\infty}\frac{1}{N}\mathbf{V}\mathbf{Z}^{\top}=\mathbf{0}$$

Choice of the instruments

The condition

$$\lim_{N\to+\infty}\frac{1}{N}\underset{ps\times h}{\mathbf{V}}\cdot\overset{h\times n_z s}{\mathbf{Z}^{\top}}=\mathbf{0}$$

- $h \coloneqq N-s+1$

implies that

$$\lim_{N\to+\infty}\frac{1}{N}\begin{bmatrix}\boldsymbol{v}_2(1) & \boldsymbol{v}_2(2) & \cdots & \boldsymbol{v}_2(N-s+1)\\ \boldsymbol{v}_2(2) & \boldsymbol{v}_2(3) & \cdots & \boldsymbol{v}_2(N-s+2)\\ \vdots & \vdots & \cdots & \vdots\\ \boldsymbol{v}_2(s) & \boldsymbol{v}_2(s+1) & \cdots & \boldsymbol{v}_2(N)\end{bmatrix}\cdot\begin{bmatrix}\boldsymbol{z}(1)^{\top}\\ \boldsymbol{z}(2)^{\top}\\ \vdots\\ \boldsymbol{z}(N-s+1)^{\top}\end{bmatrix}=\mathbf{0}$$

This can be (asymptotically) achieved if $\boldsymbol{v}_2(t)$ is **uncorrelated** with $\boldsymbol{z}(t)$

Choice of the instruments

In the open-loop case, since $u(t) \perp v_2(t)$, and due to the whiteness of $\boldsymbol{v}_2(t)$, a suitable instrumental variable is

$$\boldsymbol{z}(t)=[\boldsymbol{y}(t-1)\quad\cdots\quad\boldsymbol{y}(t-s_1)\quad\boldsymbol{u}(t-1)\quad\cdots\quad\boldsymbol{u}(t-s_2)]^{\top}$$

Usually, $s_1=s_2=s$ and so $\boldsymbol{z}(t)\in\mathbb{R}^{(p+m_u)s\times 1}$, so that $n_z=p+m_u$

This choice of $\mathbf{Z}$ constructs the instruments matrix by using **past inputs** and **past outputs**. One way to practically construct $\mathbf{Z}$ is to **split the data** into two parts: the **«past data»** and the **«future data»**

Choice of the instruments

«Past outputs»

$$\underset{ps\times(N/2-s+1)}{\mathbf{Y}_p}\coloneqq[\overrightarrow{\boldsymbol{y}_s}(1)\quad\cdots\quad\overrightarrow{\boldsymbol{y}_s}(N/2-s+1)]=\begin{bmatrix}\boldsymbol{y}(1) & \boldsymbol{y}(2) & \cdots & \boldsymbol{y}(N/2-s+1)\\ \boldsymbol{y}(2) & \boldsymbol{y}(3) & \cdots & \boldsymbol{y}(N/2-s+2)\\ \vdots & \vdots & \cdots & \vdots\\ \boldsymbol{y}(s) & \boldsymbol{y}(s+1) & \cdots & \boldsymbol{y}(N/2)\end{bmatrix}$$

«Future outputs»

$$\underset{ps\times(N/2-s+1)}{\mathbf{Y}_f}\coloneqq[\overrightarrow{\boldsymbol{y}_s}(N/2+1)\quad\cdots\quad\overrightarrow{\boldsymbol{y}_s}(N-s+1)]=\begin{bmatrix}\boldsymbol{y}(N/2+1) & \boldsymbol{y}(N/2+2) & \cdots & \boldsymbol{y}(N-s+1)\\ \boldsymbol{y}(N/2+2) & \boldsymbol{y}(N/2+3) & \cdots & \boldsymbol{y}(N-s+2)\\ \vdots & \vdots & \cdots & \vdots\\ \boldsymbol{y}(N/2+s+1) & \boldsymbol{y}(N/2+s+2) & \cdots & \boldsymbol{y}(N)\end{bmatrix}$$

Choice of the instruments

«Past inputs»

$$\underset{ps \times (N/2-s+1)}{\mathbf{U}_p} \coloneqq [\overrightarrow{\boldsymbol{u}_s}(1) \quad \cdots \quad \overrightarrow{\boldsymbol{u}_s}(N/2-s+1)] = \begin{bmatrix} \boldsymbol{u}(1) & \boldsymbol{u}(2) & \cdots & \boldsymbol{u}(N/2-s+1) \\ \boldsymbol{u}(2) & \boldsymbol{u}(3) & \cdots & \boldsymbol{u}(N/2-s+2) \\ \vdots & \vdots & \cdots & \vdots \\ \boldsymbol{u}(s) & \boldsymbol{u}(s+1) & \cdots & \boldsymbol{u}(N/2) \end{bmatrix}$$

«Future inputs»

$$\underset{ps \times (N/2-s+1)}{\mathbf{U}_f} \coloneqq [\overrightarrow{\boldsymbol{u}_s}(N/2+1) \quad \cdots \quad \overrightarrow{\boldsymbol{u}_s}(N-s+1)] = \begin{bmatrix} \boldsymbol{u}(N/2+1) & \boldsymbol{u}(N/2+2) & \cdots & \boldsymbol{u}(N-s+1) \\ \boldsymbol{u}(N/2+2) & \boldsymbol{u}(N/2+3) & \cdots & \boldsymbol{u}(N-s+2) \\ \vdots & \vdots & \cdots & \vdots \\ \boldsymbol{u}(N/2+s+1) & \boldsymbol{u}(N/2+s+2) & \cdots & \boldsymbol{u}(N) \end{bmatrix}$$

MOESP for stochastic systems (MOESP-PO)

MOESP-PO algorithm

- Collect noisy data $\mathcal{D} = \{\boldsymbol{u}(t), \boldsymbol{y}(t)\}_{t=1}^{N}$
- Build matrices $\mathbf{Y}_p, \mathbf{Y}_f, \mathbf{U}_p, \mathbf{U}_f$. Build $\mathbf{Z} = \begin{bmatrix} \mathbf{U}_p \\ \mathbf{Y}_p \end{bmatrix}$
- Perform RQ factorization of $\begin{bmatrix} \mathbf{U}_f \\ \mathbf{Z} \\ \mathbf{Y}_f \end{bmatrix} = \begin{bmatrix} R_{11} & 0 & 0 & 0 \\ R_{21} & R_{22} & 0 & 0 \\ R_{31} & R_{32} & R_{33} & 0 \end{bmatrix} \begin{bmatrix} Q_{11} \\ Q_{21} \\ Q_{31} \\ Q_4 \end{bmatrix}$
- Compute SVD of $R_{32} = U\Sigma V^\top$. Choose order n and compute the rank-n approximation $R_{32} \approx U_n \Sigma_n V_n^\top$, where $U_n = U(:, 1:n)$, $\Sigma_n = \Sigma(1:n, 1:n)$, $V_n = V(:, 1:n)$

MOESP for stochastic systems (MOESP-PO)

MOESP-PO algorithm

- Compute the estimates

$$\hat{C} = C_T = U_n(1:p, :) \qquad \hat{A} = A_T = U_n(1:(s-1)p, :)^\dagger \cdot U_n(p+1:sp, :)$$

- Solve the Least Squares problem

$$\boldsymbol{y}(t) = \Phi^\top(t) \cdot \boldsymbol{\theta} = \begin{bmatrix} \sum_{\tau=0}^{t-1} \boldsymbol{u}(\tau)^\top \otimes C_T A_T^{t-\tau-1} & \boldsymbol{u}(t)^\top \otimes I_p & CA_T^t \end{bmatrix} \begin{bmatrix} \mathrm{vec}(B_T) \\ \mathrm{vec}(D_T) \\ \boldsymbol{x}_T(0) \end{bmatrix}$$

$$\hat{\boldsymbol{\theta}} = \left(\sum_{t=0}^{N-1} \Phi(t) \cdot \Phi(t)^\top \right)^{-1} \cdot \sum_{t=0}^{N-1} \Phi(t) \cdot \boldsymbol{y}(t)$$

UNIVERSITÀ DEGLI STUDI DI BERGAMO | Dipartimento di Ingegneria Gestionale, dell'Informazione e della Produzione

UNIVERSITÀ DEGLI STUDI DI BERGAMO | Dipartimento di Ingegneria Gestionale, dell'Informazione e della Produzione

ADAPTIVE LEARNING, ESTIMATION AND SUPERVISION OF DYNAMICAL SYSTEMS (ALES)

Lecture 9: Introduction to supervision of dynamical systems

Master Degree in COMPUTER ENGINEERING

Data Science and Data Engineering Curriculum

SPEAKER
Prof. Mirko Mazzoleni

PLACE
University of Bergamo

Syllabus

1. **Recursive and adaptive identification**
   - 1.1 Recursive ARX estimation (RLS)
   - 1.2 Least Mean Squares (LMS)
   - 1.3 Instrumental Variables (IV)
2. **Closed-loop identification**
3. **Subspace and MIMO identification**
   - 3.1 Singular Value Decomposition
   - 3.2 Impulse data: Ho-Kalman, Kung algorithms
   - 3.3 Generic I/O data: the MOESP algorithm
4. **Supervision of dynamical systems**
   - 4.1 Introduction to fault diagnosis
   - 4.2 Model-based fault diagnosis
   - 4.3 Parity space approaches
   - 4.4 Observer-based approaches
   - 4.5 Signal-based fault diagnosis
   - 4.6 Knowledge-based fault diagnosis

CASE STUDIES

- Virtual Reference Feedback Tuning
- Nuclear particles classification
- Leak detection in an industrial valve
- Bearing fault identification

Outline

1. Supervision and diagnosis of industrial plants
2. Supervision aims
3. Supervision methodologies
4. Fault detection with limit-checking

Outline

1. **Supervision and diagnosis of industrial plants**
2. Supervision aims
3. Supervision methodologies
4. Fault detection with limit-checking

Supervision and diagnosis of industrial plants

Supervision tasks aim to discover **undesired** or unpermitted **states** of the system and taking appropriate **corrective actions** to avoid damages or accidents

Undesired and unpermitted states are caused by **faults**. Faults can lead to **failures**

Definition (fault)

A **fault** is an **unpermitted deviation** of at least one characteristic property of the system (process/component) from the acceptable, usual, **standard condition.** It is a **state**

Definition (failure)

A **failure** is a permanent or temporary **interruption** of a system's ability to perform a required **function** under **specified operating conditions**. It is an **event**

Supervision and diagnosis of industrial plants

A **fault** in a process or component can appear due to **internal** or **external** causes

Examples of external causes are:

- **environmental factors** as *humidity*, *dust*, *chemicals*, *electromagnetic radiation*, *temperature*, leading to e.g. corrosion or pollution

Examples of internal causes are:

- **missing lubrication** leading to higher *friction* or *wear*
- **overheating**
- **leaks**
- **short-circuits**

Supervision and diagnosis of industrial plants

System	Fault	Failure	Failure effect on component	Failure effect on system
Electrical illumination	Switch with corroded contacts	Switch not performing switching	Occasionally no electrical conductivity	Intermittent light
	Broken wire in cable	Wire not transmitting current	No electrical conductivity	No light
DC motor	Worn brushes	Brushes not generating torque well	Varying electrical flux	Varying torque and speed
	Broken wire in excitation coil	Brushes not generating torque at all	No electrical flux	No torque, no speed
Machine tool belt drive	Belt wit too low pretension	Belt not transmitting torque well	-	Sluggish dynamic, piecewise motion
	Broken belt threads	Belt not transmitting torque	-	Standstill of feed drive

Supervision and diagnosis of industrial plants

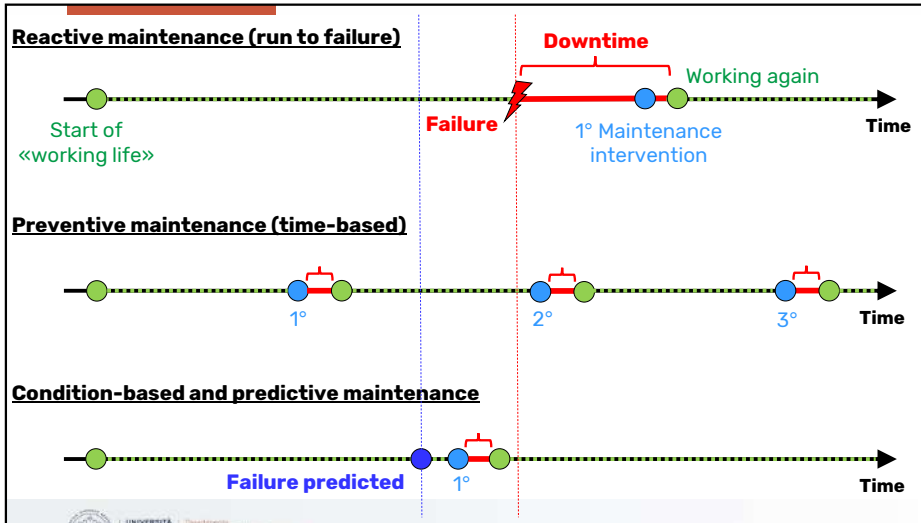
System	Fault	Failure	Failure effect on component	Failure effect on system
Pneumatic valve	Leak in supply air conduct	Valve not performing well	Slow shaft motion, limited position range	Closed-loop system does not follow the setpoint for some time
	Corroded shaft	Valve not performing	Mechanical friction too high	No motion, permanent control deviation

Supervision and diagnosis of industrial plants

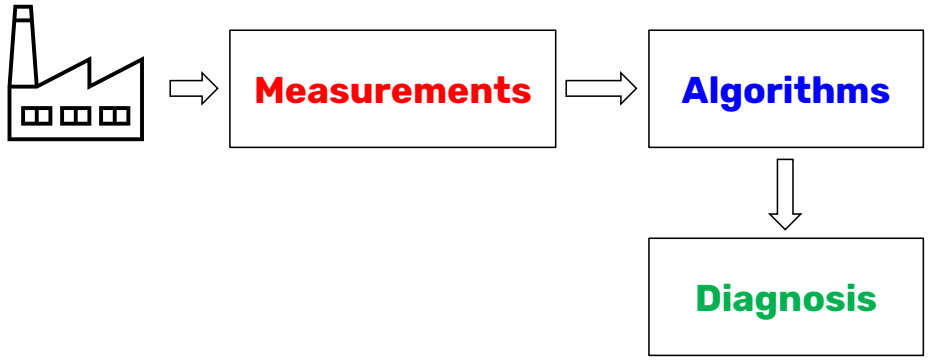
Supervision tasks are a key component of **industry 4.0.** Their main purpose is to:

Optimize maintenance **interventions** and reduce **costs**

1. **Reactive** maintenance *(run-to-failure)*
2. **Preventive** maintenance *(time-based)*
3. **Condition-based** maintenance
4. **Predictive** maintenance



Basic idea



Signals thresholding (tolerances checks)

Most supervision of technical engineering processes, or the quality control of products in manufacturing, is performed by means of **limit-checking** or **signal thresholding** of some measurable **variables** $y(t)$ as:

- Pressures
- Liquid levels
- Currents
- Forces
- Temperatures
- Speeds
- Oscillations (positions)

The basic idea is to check if those quantities are **within a tolerance zone** as

$$y_{\min} \leq y(t) \leq y_{\max}$$

An **alarm** is triggered if the tolerance zone is **exceeded**

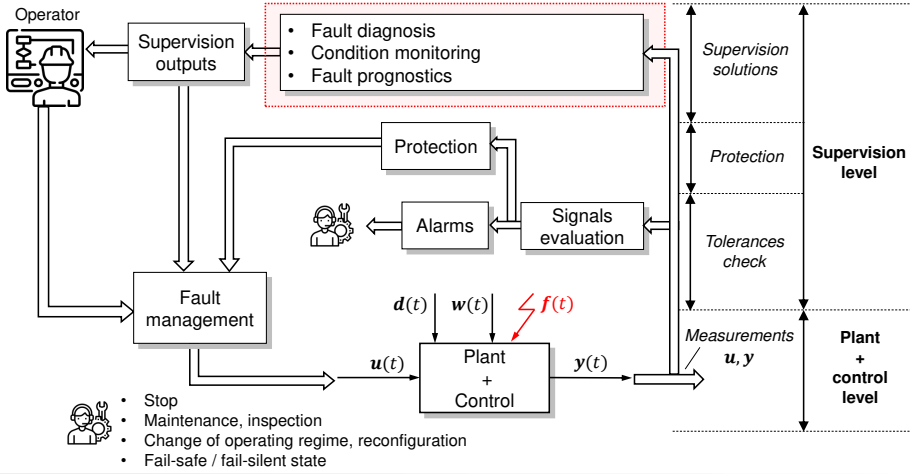
Signals thresholding (tolerances checks)

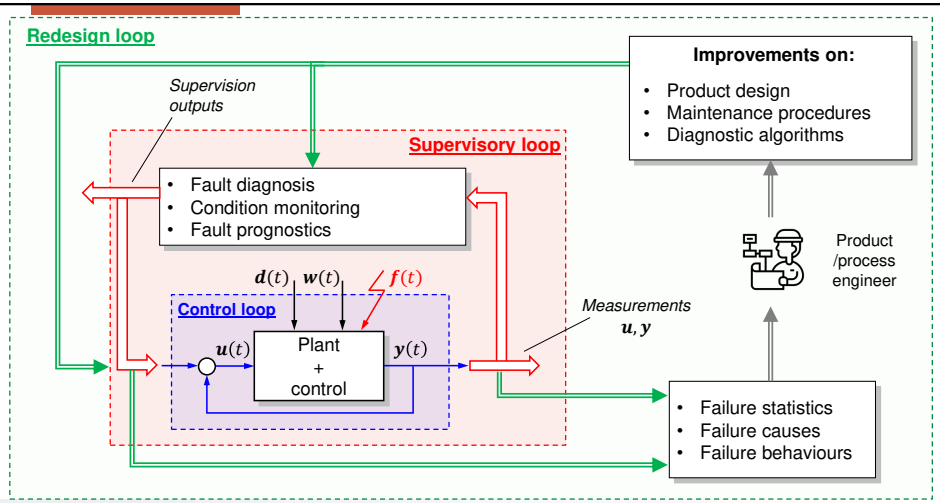
Limit checking with **fixed thresholds** works well if the process is in **steady state** or the monitored variables do not depend on the **process operating point**

For large-scale processes with many monitored and limit-checked variables, it may happen that, after a severe process fault or failure, **several alarms** may be triggered in a **short time** sequence, generating an **«alarm shower»**

The operator is then **overloaded** with information and it is more difficult to think an immediate **reaction** and find the **root cause** of the problem

Thus, more **advanced diagnostic algorithms** for supervision are needed





Supervision algorithms

Supervision algorithms can have different **aims:**

- **Fault diagnosis:** *dichotomous* (healthy/faulty, other info) output at each *actual* time
- **Condition monitoring:** *continuous* output (health indicator) at each *actual* time
- **Fault prognostics:** *continuous* output (health indicator) at *future* times

Supervision algorithms can be implemented using different **rationales:**

- **Model-based:** relies on an *input-output model* of the plant or active component
- **Signal-based:** relies on specific indicators *(symptoms)* computed on specific signals
- **Knowledge-based:** relies only on *historical* healthy (and possibly faulty) *data*

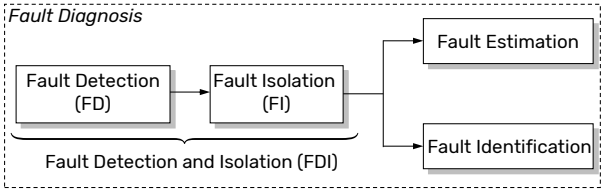
Outline

1. Supervision and diagnosis of industrial plants
2. **Supervision aims**
3. Supervision methodologies
4. Fault detection with limit-checking

Supervision aims: fault diagnosis

The aim of **fault diagnosis** is to discover **anomalous behaviours** in engineering plants

1. **Fault detection (FD):** detect if there is an anomaly *(fault present/not present)*
2. **Fault isolation:** detect the faulty component *(fault detection + fault location)*
3. **Fault identification:** type, size (severity) and nature (cause) of detected faults
4. **Fault estimation:** estimate a fault «signal»



Supervision aims: condition monitoring

Condition monitoring refers to the continued oversight of the progressive degradation of the monitored plant or component

The aim is to generate one or more **indicators** of health state that monotonically **evolve** as plant/component degradation **progresses**

The output of the method is not a fault presence/not presence decision, but also a **time evolution** of the health state

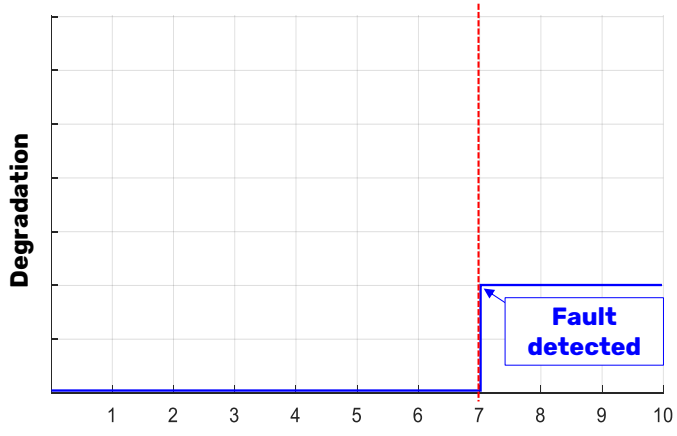
Supervision aims: fault prognostics

Fault prognostics collects the **continuous indications** of the current health state of the system (provided by the condition monitoring function), to **forecast** their future evolution and estimate the **Remaining Useful Life (RUL)** of the component

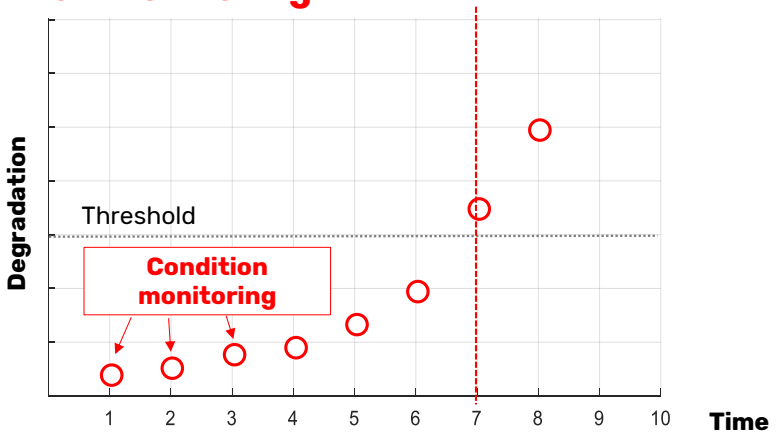
Fault prognostics requires the development of a **model of the fault evolution**, to extrapolate its future trend

Fault prognostics is an **iterative procedure**, since the predictions have to be updated every time a new «health state point» is available from the condition monitoring

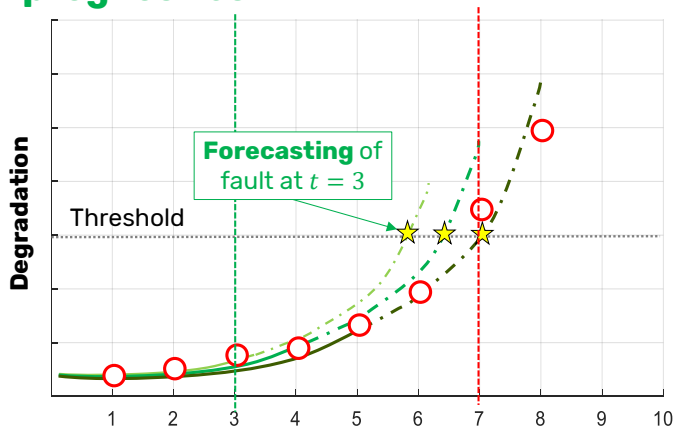
1. Fault detection



2. Condition monitoring



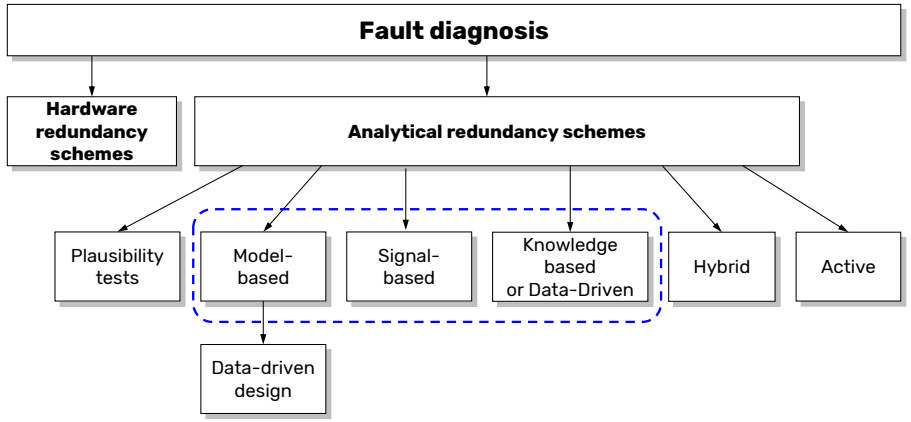
3. Fault prognostics



Outline

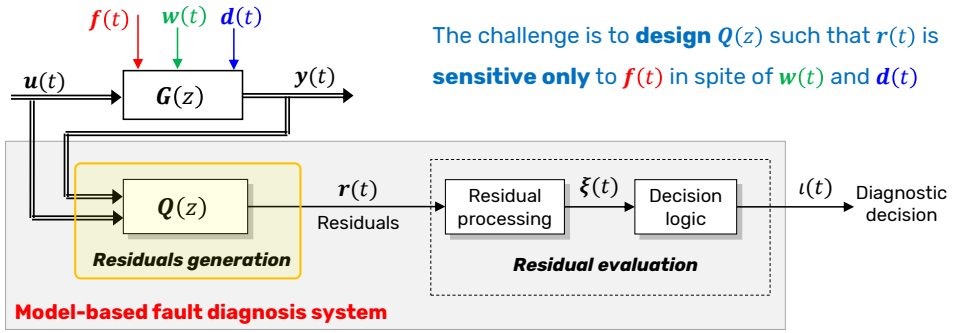
1. Supervision and diagnosis of industrial plants
2. Supervision aims
3. **Supervision methodologies**
4. Fault detection with limit-checking

Diagnostic methodologies



Model-based fault diagnosis

A model of the system is developed to design a **residual generator** $Q(z)$



The challenge is to **design** $Q(z)$ such that $r(t)$ is **sensitive only** to $f(t)$ in spite of $w(t)$ and $d(t)$

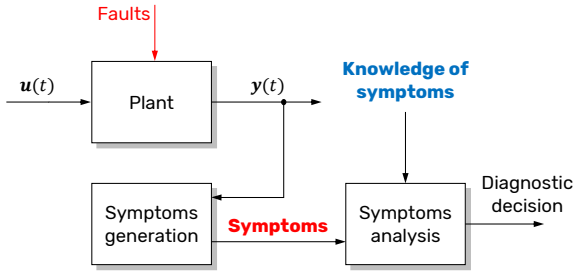
Signal-based fault diagnosis

Certain process signals carry information about the faults to be detected

- From them, **fault symptoms** are computed…
- …and compared with **prior knowledge about the faults**

Examples are:

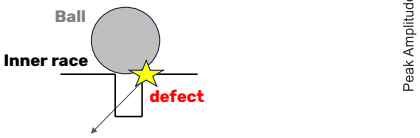
- Diagnosis of rotating components defects
- Stator\rotor current faults



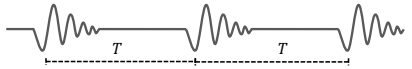
Example: rolling bearings diagnosis

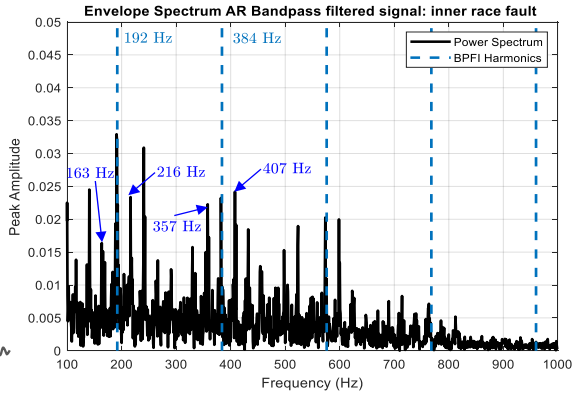
BallPass Frequency Inner race

$$\mathrm{BPFI} = \frac{nf_r}{2}\left\{1 + \frac{d}{D}\cos\phi\right\}$$



Vibration due to the impulsive «clash» with a localized defect





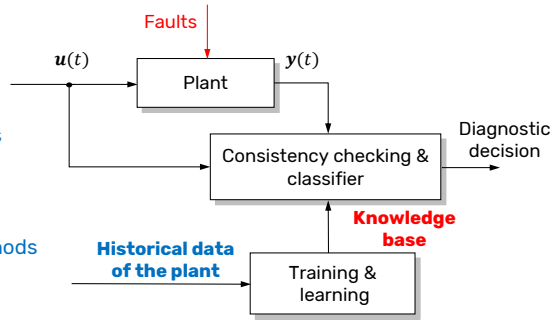
Knowledge-based fault diagnosis

Great amount of historical data available:

- There is **not a prior knowledge** about the faults, and it has to be **extracted from data**
- **Training vs testing** phases

Common approaches are:

- Machine learning supervised classifiers
- Unsupervised anomaly detection
- Transfer learning
- Statistical Process Control (SPC) methods (control charts, PCA indicators,…)



Outline

1. Supervision and diagnosis of industrial plants
2. Supervision aims
3. Supervision methodologies
4. **Fault detection with limit-checking**

Fault detection with limit-checking

As previously stated, **limit-checking** or **signal-thresholding** consists in comparing measured variables $y(t)$, or their first derivative $\dot{y}(t)$, with certain **thresholds**

We may then have:

- Limit-checking of the signals **raw value**
- Limit-checking of the signals **first derivative** *(trend-checking)*
- **Combined** raw value – trend checking

Generally, **two limits values** (thresholds) are set:

- an **upper** threshold
- a **lower** threshold

Limit-checking of the signals raw values

With limit-checking of signals **raw values**, a **normal state** is present when process variables are inside the following **tolerance zone**

$$y_{\min} \leq y(t) \leq y_{\max}$$

where $y_{\min}$ and $y_{\max}$ denote the lower and upper thresholds, respectively. There is a **trade-off** between **false alarms** and fault **detection** capability

Exceeding one of the thresholds indicate a **fault** somewhere in the process. Examples of applications of this method are:

- **Oil pressure** *(lower limit)*
- **Control error** of a control loop *(higher limit)*
- **Coolant water temperature** of combustion engines *(higher limit)*

Limit-checking of the signals trend

With limit-checking of signals **first derivative (trend)**, a **normal state** is present when process variables are inside the following **tolerance zone**

$$\dot{y}_{\min} \leq \dot{y}(t) \leq \dot{y}_{\max}$$

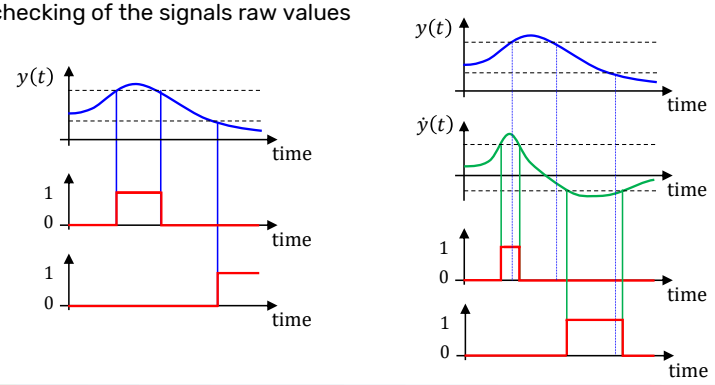
where $\dot{y}_{\min}$ and $\dot{y}_{\max}$ denote the lower and upper thresholds, respectively. Again, there is a **trade-off** between **false alarms** and fault **detection** capability

Trend checking is applied, for instance, to

- **Oil pressures** *(lower limit)* and **vibrations** *(higher limit)* of oil bearings in turbines
- **Wear measures** (surface texture of machined workpiece, acoustics, vibration, feed forces, and current consumption) of machines *(higher limit)*

Limit-checking of the signals trend

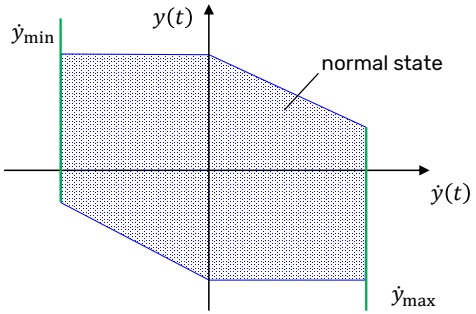
If **relatively small thresholds** are selected, an alarm can be obtained **earlier** than for limit-checking of the signals raw values



Combined limit-checking

Combination of raw values and trend limit-checking can be performed by making the **threshold** of raw values **dependent** on the trend, to:

- **detect** fast developing deviations **early** *(lower raw values thresholds)*
- **avoid false alarms** for small trends *(higher raw values thresholds)*



Signal prediction for limit-checking

A further improvement over trend limit-checking can be realized using **signal prediction**. For **deterministic signals** sampled at discrete time instants t, a polynomial of the form

$$y(t) = a_0 + a_1 t + a_2 t^2 + \cdots$$

can be recursively fit by **recursive least squares**, to **extrapolate** k steps ahead in time. Then, the predicted value is compared with thresholds

In the case of **stochastic signals**, stochastic time-series model (using Kolmogorov-Wiener theory of Kalman filter) can be fit to data and an optimal k-step ahead prediction performed

Advanced material

Change detection of statistical quantities

When the monitored quantities can be thought as stochastic variables, it is possible to monitor an estimate of their **mean** and **standard deviation** (and other statistics if desired, such as the **kurtosis**)

These estimates are again compared with thresholds. For instance, assuming a changing mean but constant variance σ_0^2, a possible threshold for the mean is

$$\mathrm{Th}_{\mathrm{mean}} = \kappa \cdot \sigma_0^2$$

where e.g. $|\kappa| \geq 2$. **Statistical hypothesis tests** could be applied here: however, the assumptions for their validity rarely holds in industrial practice. Thus, it is better to use these simple threshold rules with eventually **adaptive thresholds**

Change detection of statistical quantities

The **mean** m_0 and the **variance** σ_0^2 can be **recursively** estimated, eventually with the addition of a **forgetting factor** μ

Adaptive mean: $\hat{m}(t) = \mu \cdot \hat{m}(t-1) + (1-\mu) \cdot y(t)$

Adaptive variance: $\hat{\sigma}^2(t) = \frac{2\mu - 1}{\mu} \cdot \hat{\sigma}^2(t-1) + (1-\mu)[y(t) - \hat{m}(t-1)]^2$

The **adaptive mean** can be thought as the output of a first-order discrete **low pass filter** with $a_1 = -\mu$ and $b_0 = (1 + a_1)$, so that

$$\hat{m}(t) = -a_1 \cdot \hat{m}(t-1) + b_0 \cdot y(t)$$

Change detection of statistical quantities

The variance can be expressed as

$$\sigma_0^2(t) = \mathbb{E}[y^2(t)] - \mathbb{E}[y(t)]^2$$

where also $y_f^2(t) \coloneqq \mathbb{E}[y^2(t)]$ can be thought as the output of the previous filter with input $y^2(t)$, so that an estimate $\hat{y}_f^2(t)$ is obtained as

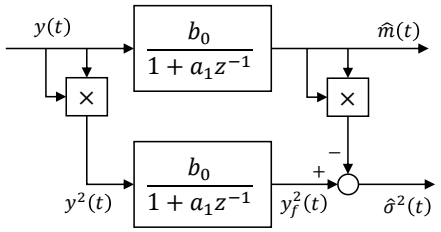
$$\hat{y}_f^2(t) = -a_1 \cdot \hat{y}_f^2(t) + b_0 \cdot y^2(t)$$

So, the **adaptive variance estimate** can be computed as

$$\hat{\sigma}^2(t) = \hat{y}_f^2(t) - \hat{m}^2(t)$$

Change detection of statistical quantities

Thus, adaptive mean and variance estimate with **forgetting factor** μ can be recursively computed with the following **filtering** scheme

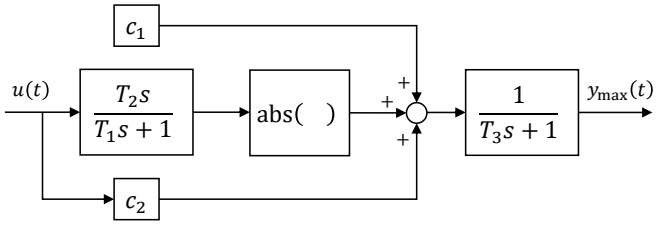


- $b_0 = 1 - \mu$
- $a_1 = -\mu$

Adaptive thresholds

Adaptive thresholds have been proposed in order to deal with **disturbances** (and model uncertainties) affecting the measured signals, so that to **reduce false alarms**

A schematic for a signal-based adaptive threshold is as follows, where $u(t)$ denotes a generic input signal (e.g. the adaptive mean estimate or the input of a dyamical system)



- T_1 and T_3 depend on the dominating time constant of the process
- T_2/T_1 depends on the uncertainty level

Advanced material

Plausibility checks

A rough supervision of measured variables is sometimes performed by checking the **plausibility** of its indicated values. Basically, it is a way to **combine limit-checking** of **several measurements**, with logic rules like

$$\text{IF } (y_{1,\min} < y_1(t) < y_{1,\max}) \text{ THEN } (y_{2,\min} < y_2(t) < y_{2,\max})$$

For instance, consider the oil pressure $p_{\text{oil}}(t)$ of a combustion engine with motor speed $\omega_{\text{motor}}(t)$ and cooling water temperature $T_{\text{water}}(t)$. We might have something like:

$$\text{IF } (\omega_{\text{motor}}(t) < 1500 \text{ rpm}) \text{ AND } (T_{\text{water}} < 50° \text{ C}) \text{ THEN } (3 \text{ bar} < p_{\text{oil}}(t) < 5 \text{ bar})$$

Plausibility checks

These rules and their values allow a rough description of the process expected behaviour under certain **healthy operating conditions** and represent a rough **process model**

If the ranges of the variables are made increasingly smaller, many rules would be necessary to describe all their interactions. In this case, it is better to use **dynamical models** of the process

Thus, plausibility checks can be seen as a first step towards **model-based** fault detection methods

Fuzzy plausibility checks

Plausibility checks can also be performed using **fuzzy logic**, has an extension of classical (binary) logic. Fuzzy inference systems (FIS) of this type are called *Mamdami FIS*

In fuzzy logic, the range of each variable is divided in continuous (possibly overlapping) portions of values, where to each portion is assigned a **membership function**, specifying the **membership degree** of the input to a certain **membership value**

Then, the **fuzzy rules** are evaluated according to their **activation degree** given current inputs, and combined toghether with logical and geometrical operators

The **output is a real number** representing a certain quantity. Fuzzy inference systems can also be used as **universal approximators** for functions *(Takagi-Sugeno FIS)*

Advanced material

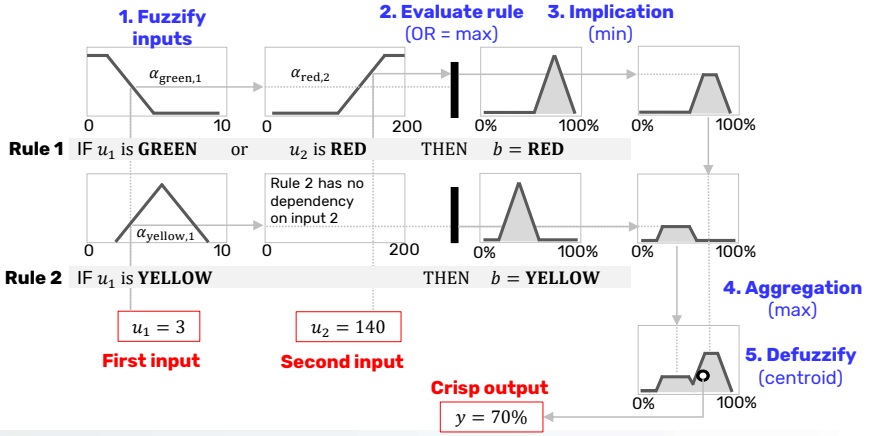
Fuzzy plausibility checks

Fuzzy rules and membership function values and shapes can be manually **defined by the user** using **prior knowledge** about the process variables. This is mostly done for Mamdani FIS

Alternatively, when input-output data are available and the aim is to **learn a nonlinear mapping** from inputs to outputs, Takagi-Sugeno FIS can be **learnt from I/O data** (i.e. learning the rules and the values of some fixed-type membership functions)

In Matlab, the `fuzzy inference toolbox` and `anfis` method can be used to design fuzzy inference systems

Fuzzy plausibility checks



UNIVERSITÀ DEGLI STUDI DI BERGAMO Dipartimento di Ingegneria Gestionale, dell'Informazione e della Produzione

ADAPTIVE LEARNING, ESTIMATION AND SUPERVISION OF DYNAMICAL SYSTEMS (ALES)

Lecture 10: Model-based fault diagnosis

Master Degree in COMPUTER ENGINEERING

Data Science and Data Engineering Curriculum

SPEAKER
Prof. Mirko Mazzoleni

PLACE
University of Bergamo

Syllabus

1. **Recursive and adaptive identification**
   - 1.1 Recursive ARX estimation (RLS)
   - 1.2 Least Mean Squares (LMS)
   - 1.3 Instrumental Variables (IV)
2. **Closed-loop identification**
3. **Subspace and MIMO identification**
   - 3.1 Singular Value Decomposition
   - 3.2 Impulse data: Ho-Kalman, Kung algorithms
   - 3.3 Generic I/O data: the MOESP algorithm
4. **Supervision of dynamical systems**
   - 4.1 Introduction to fault diagnosis
   - 4.2 Model-based fault diagnosis
   - 4.3 Parity space approaches
   - 4.4 Observer-based approaches
   - 4.5 Signal-based fault diagnosis
   - 4.6 Knowledge-based fault diagnosis

CASE STUDIES

- Virtual Reference Feedback Tuning
- Nuclear particles classification
- Leak detection in an industrial valve
- Bearing fault identification

Outline

1. The model-based fault diagnosis framework
2. Modeling of faulty systems
3. Residuals generation

Outline

1. **The model-based fault diagnosis framework**
2. Modeling of faulty systems
3. Residuals generation

The model-based fault diagnosis framework

Model-based fault detection methods are based on the idea of **analytical redundancy**, i.e. the comparison of the **actual** behaviour of the monitored plant with the behaviour predicted by a **mathematical plant model**

Plant **observations** (inputs and outputs) are **checked for consistency** with the mathematical model

The outcome of the consistency checks are quantities called **residuals**. Residuals are nominally zero, and become nonzero in presence of **fault**, **disturbances**, **noise** and **modeling error**

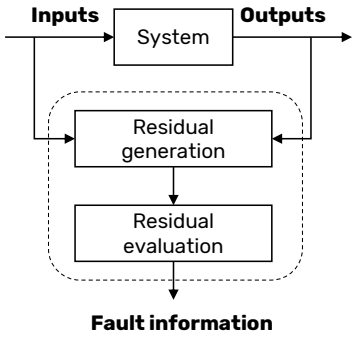
The model-based fault diagnosis framework

Residuals are then analyzed to arrive at a **diagnostic decision**

Any diagnostic algorithm which utilizes analytical redundancy consists of two blocks: the **residual generator** and the **residual evaluator**

While a single residual can be sufficient for fault detection, the **isolation** of faults (generally) requires a **set of residuals**

These residuals are usually **enhanced**, i.e. generated with specific isolation properties



Faults classification and representation

Faults can be described by

- Temporal behaviour
- Physical location
- Effect on plant and instrumentation

CLASSIFICATION BY TEMPORAL BEHAVIOUR

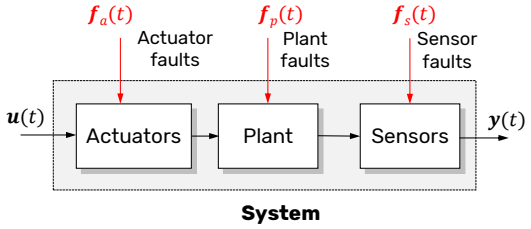
It concerns the **time scales** characterizing the transition from normal to faulty behaviour:

- **persistent faults** have a **long-range time evolution**, and may then arise **abruptly** (step-wise) or **incipiently** (ramp-wise). The diagnosis of incipient faults is the main reason for condition monitoring
- **intermittent faults** occur only for a short time interval, disappear, and then appear again, repeatedly (impulse-wise)

Faults classification and representation

CLASSIFICATION BY PHYSICAL LOCATION

- **Actuator faults:** discrepancies between the **input command** of an actuator and its **actual output**
- **Sensor faults:** discrepancies between the **measured** and **actual values** of individual plant variables
- **Plant faults:** **changes** of **some plant parameters** or **unknown inputs** acting on the plant who change the plant output independently of its inputs



Faults classification and representation

CLASSIFICATION BY EFFECT

The effect induced by a fault can be modeled as:

- Additional **unknown inputs** acting on plant and instrumentation *(additive faults)*
- **Changes** of some plant **parameters** *(multiplicative faults)*

Sometimes the classification of a fault in these two types follows from its nature; sometimes, it is **arbitrary**. Usually, faults are modeled as **additive** when possible, since they are **simpler to treat**

Disturbances and noises

A **disturbance** is, like additive faults, an unknown **deterministic** extra input acting on the plant. The difference is subjective: **faults** are inputs that have **to be detected**, while **disturbances** have **to be ignored** and should not affect the diagnostic decision

Noises are unknown extra inputs like disturbances, but they are usually assumed to exhibit **random behaviour**

It is possible to consider as a **disturbance** an input which effect can be **completely rejected** (decoupled) from the residuals, while the **noises** effects on them can only be **attenuated**

Modeling uncertainties

Modeling uncertainties are errors or uncertainties in the **parameters** of a plant model

Just like multiplicative faults, they are discrepancies between the true plant and its model, but they represent an **interference** for the diagnosis

Like disturbances and noises, their effect on residuals should be **minimized**

Thus, modeling uncertainties can be considered as **multiplicative disturbances**

	Additive	Multiplicative
Faults	sensor fault actuator fault plant fault	plant fault
Disturbances	plant disturbance	modeling error
Noises	sensor noise actuator noise plant noise	

Outline

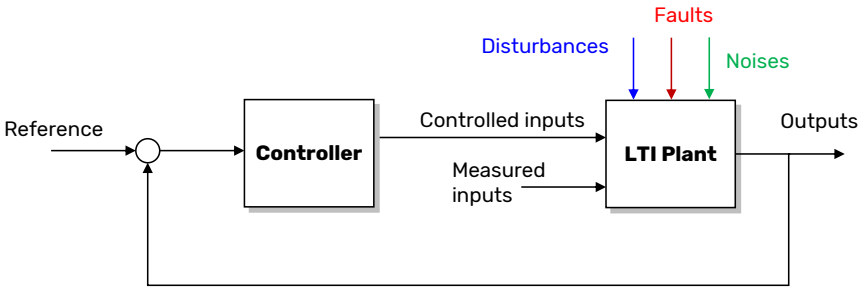
1. The model-based fault diagnosis framework

2. **Modeling of faulty systems**

3. Residuals generation

Modeling of faulty systems

Consider a generic closed-loop system

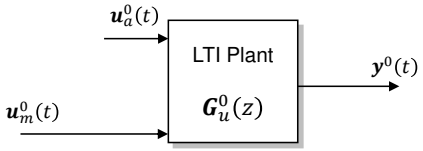


Modeling of faulty systems

Consider a system without faults, disturbances and noise acting on it. In absence of those exogenous signals, the **nominal input-output relationship** is

$$\underset{p\times 1}{\boldsymbol{y}^0(t)} = \underset{p\times m_u}{\boldsymbol{G}_u^0(z)}\underset{m_u\times 1}{\boldsymbol{u}^0(t)}$$

$$= \underset{p\times m_{u_a}}{[\boldsymbol{G}_a^0(z)}\ \ \underset{p\times m_{u_m}}{\boldsymbol{G}_m^0(z)}]\underset{m_u\times 1}{\begin{bmatrix}\boldsymbol{u}_a^0(t)\\ \boldsymbol{u}_m^0(t)\end{bmatrix}}\begin{matrix}m_{u_a}\times 1\\ m_{u_m}\times 1\end{matrix}$$



- $\boldsymbol{u}_a^0(t) \in \mathbb{R}^{m_{u_a}\times 1}$: **actuated** nominal inputs
- $\boldsymbol{u}_m^0(t) \in \mathbb{R}^{m_{u_m}\times 1}$: **measured** nominal inputs
- $\boldsymbol{G}_u^0(z)$ is a $p \times m_u$ transfer functions matrix

$$m_u = m_{u_a} + m_{u_m}$$

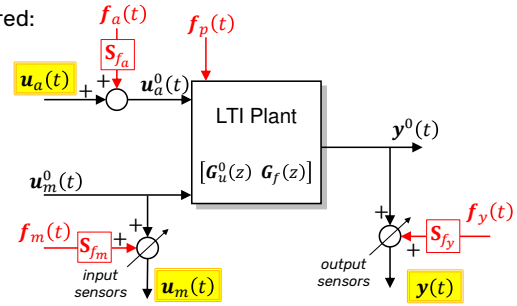
Modeling of faulty systems

If an input is **neither actuated nor measured**, then it is unknown and should be handled as a **disturbance**

Additive faults

The following **additive faults** are considered:

- Input actuator **faults** $\boldsymbol{f}_a(t) \in \mathbb{R}^{m_{f_a}\times 1}$
- Input sensor **faults** $\boldsymbol{f}_m(t) \in \mathbb{R}^{m_{f_m}\times 1}$
- Plant **faults** $\boldsymbol{f}_p(t) \in \mathbb{R}^{m_{f_p}\times 1}$
- Output sensor **faults** $\boldsymbol{f}_y(t) \in \mathbb{R}^{m_{f_y}\times 1}$



The matrices $\boldsymbol{S}_{f_a} \in \mathbb{R}^{m_{u_a}\times m_{f_a}}$, $\boldsymbol{S}_{f_m} \in \mathbb{R}^{m_{u_m}\times m_{f_m}}$, $\boldsymbol{S}_{f_y} \in \mathbb{R}^{p\times m_{f_y}}$ **distribute** the effect of faults on nominal signals

Additive faults

The **measured** variables $\boldsymbol{u}_a(t), \boldsymbol{u}_m(t), \boldsymbol{y}(t)$ are related to the **actual** plant inputs $\boldsymbol{u}_a^0(t), \boldsymbol{u}_m^0(t)$ and outputs $\boldsymbol{y}^0(t)$ as

$$\underset{m_{u_a}\times 1}{\boldsymbol{u}_a^0(t)} = \underset{m_{u_a}\times 1}{\boldsymbol{u}_a(t)} + \underset{m_{u_a}\times m_{f_a}}{\boldsymbol{S}_{f_a}}\cdot\underset{m_{f_a}\times 1}{\boldsymbol{f}_a(t)}$$

$$\underset{m_{u_m}\times 1}{\boldsymbol{u}_m^0(t)} = \underset{m_{u_m}\times 1}{\boldsymbol{u}_m(t)} - \underset{m_{u_m}\times m_{f_m}}{\boldsymbol{S}_{f_m}}\cdot\underset{m_{f_m}\times 1}{\boldsymbol{f}_m(t)}$$

$$\underset{p\times 1}{\boldsymbol{y}^0(t)} = \underset{p\times 1}{\boldsymbol{y}(t)} - \underset{p\times m_{f_y}}{\boldsymbol{S}_{f_y}}\cdot\underset{m_{f_y}\times 1}{\boldsymbol{f}_y(t)}$$

Thus, the nominal **relation** between **plant inputs and outputs** becomes

$$\underset{p\times 1}{\boldsymbol{y}^0(t)} = \underset{p\times m_u}{[\boldsymbol{G}_a^0(z)}\ \ \underset{p\times m_u}{\boldsymbol{G}_m^0(z)}]\underset{m_u\times 1}{\begin{bmatrix}\boldsymbol{u}_a^0(t)\\ \boldsymbol{u}_m^0(t)\end{bmatrix}}$$

$$\boldsymbol{y}(t) - \boldsymbol{S}_{f_y}\cdot\boldsymbol{f}_y(t) = [\boldsymbol{G}_a^0(z)\ \ \boldsymbol{G}_m^0(z)]\begin{bmatrix}\boldsymbol{u}_a(t) + \boldsymbol{S}_{f_a}\cdot\boldsymbol{f}_a(t)\\ \boldsymbol{u}_m(t) - \boldsymbol{S}_{f_m}\cdot\boldsymbol{f}_m(t)\end{bmatrix} + \underset{p\times m_{f_p}}{\boldsymbol{G}_{f_p}(z)}\underset{m_{f_p}\times 1}{\boldsymbol{f}_p(t)}$$

Additive faults

Computing the products

$$\boldsymbol{y}(t) = \boldsymbol{G}_a^0(z)\boldsymbol{u}_a(t) + \boldsymbol{G}_a^0(z)\boldsymbol{S}_{f_a}\boldsymbol{f}_a(t) + \boldsymbol{G}_m^0(z)\boldsymbol{u}_m(t) - \boldsymbol{G}_m^0(z)\boldsymbol{S}_{f_m}\boldsymbol{f}_m(t) + \boldsymbol{G}_{f_p}(z)\boldsymbol{f}_p(t) + \boldsymbol{S}_{f_y}\boldsymbol{f}_y(t)$$

we can write the relations in the more compact form:

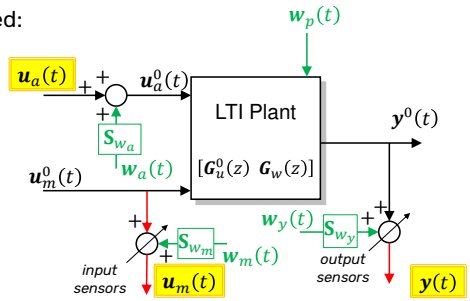
$$\underset{p\times 1}{\boldsymbol{y}(t)} = \underset{p\times m_u}{\boldsymbol{G}_u^0(z)}\underset{m_u\times 1}{\boldsymbol{u}(t)} + \underset{p\times m_f}{\boldsymbol{G}_f(z)}\underset{m_f\times 1}{\boldsymbol{f}(t)}$$

- $\underset{p\times m_f}{\boldsymbol{G}_f(z)} = \left[\underset{p\times m_{f_a}}{\boldsymbol{G}_a^0(z)\boldsymbol{S}_{f_a}} \quad \underset{p\times m_{f_m}}{\boldsymbol{G}_m^0(z)\boldsymbol{S}_{f_m}} \quad \underset{p\times m_{f_p}}{\boldsymbol{G}_{f_p}(z)} \quad \underset{p\times m_{f_y}}{\boldsymbol{S}_{f_y}}\right]$
- $\underset{m_f\times 1}{\boldsymbol{f}(t)} = \left[\underset{1\times m_{f_a}}{\boldsymbol{f}_a^\top(t)} \quad \underset{1\times m_{f_m}}{-\boldsymbol{f}_m^\top(t)} \quad \underset{1\times m_{f_p}}{\boldsymbol{f}_p^\top(t)} \quad \underset{1\times m_{f_y}}{\boldsymbol{f}_y^\top(t)}\right]^\top$

Additive noises

The following **additive noises** are considered:

- Input actuator **noises** $\boldsymbol{w}_a(t) \in \mathbb{R}^{m_{w_a}\times 1}$
- Input sensor **noises** $\boldsymbol{w}_m(t) \in \mathbb{R}^{m_{w_m}\times 1}$
- Plant **noises** $\boldsymbol{w}_p(t) \in \mathbb{R}^{m_{w_p}\times 1}$
- Output sensor **noises** $\boldsymbol{w}_y(t) \in \mathbb{R}^{m_{w_y}\times 1}$



The matrices $\boldsymbol{S}_{w_a} \in \mathbb{R}^{m_{u_a}\times m_{w_a}}$, $\boldsymbol{S}_{w_m} \in \mathbb{R}^{m_{u_m}\times m_{w_m}}$, $\boldsymbol{S}_{w_y} \in \mathbb{R}^{p\times m_{w_y}}$ distribute the effect of noises on nominal signals

Additive noises

Analogous to faults, we have that

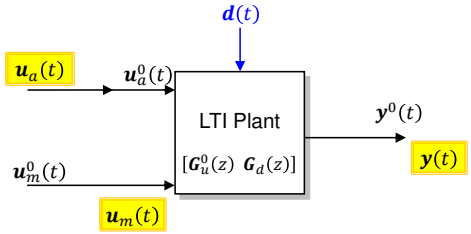
$$\underset{p\times 1}{\boldsymbol{y}(t)} = \underset{p\times m_u}{\boldsymbol{G}_u^0(z)}\underset{m_u\times 1}{\boldsymbol{u}(t)} + \underset{p\times m_w}{\boldsymbol{G}_w(z)}\underset{m_w\times 1}{\boldsymbol{w}(t)}$$

- $\underset{p\times m_w}{\boldsymbol{G}_w(z)} = \left[\underset{p\times m_{w_a}}{\boldsymbol{G}_a^0(z)\boldsymbol{S}_{w_a}} \quad \underset{p\times m_{w_m}}{\boldsymbol{G}_m^0(z)\boldsymbol{S}_{w_m}} \quad \underset{p\times m_{w_p}}{\boldsymbol{G}_{w_p}(z)} \quad \underset{p\times m_{w_y}}{\boldsymbol{S}_{w_y}}\right]$ maps the noises to the outputs
- $\underset{m_w\times 1}{\boldsymbol{w}(t)} = \left[\underset{1\times m_{w_a}}{\boldsymbol{w}_a^\top(t)} \quad \underset{1\times m_{w_m}}{-\boldsymbol{w}_m^\top(t)} \quad \underset{1\times m_{w_p}}{\boldsymbol{w}_p^\top(t)} \quad \underset{1\times m_{w_y}}{\boldsymbol{w}_y^\top(t)}\right]^\top$

Additive disturbances

The following **additive disturbances** are considered:

- Plant **disturbances** $\boldsymbol{d}(t) \in \mathbb{R}^{m_d\times 1}$

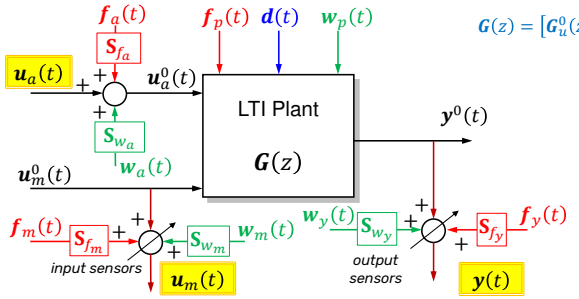


So that

$$\underset{p\times 1}{\boldsymbol{y}(t)} = \underset{p\times m_u}{\boldsymbol{G}_u^0(z)}\underset{m_u\times 1}{\boldsymbol{u}(t)} + \underset{p\times m_d}{\boldsymbol{G}_d(z)}\underset{m_d\times 1}{\boldsymbol{d}(t)}$$

Where $\boldsymbol{G}_d(z)$ maps the disturbances to the outputs

Overall additive signals representation

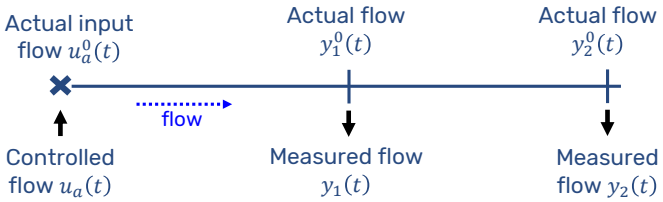


$$G(z) = [G_u^0(z) \; G_d(z) \; G_w(z) \; G_f(z)]$$

$$\underset{p\times 1}{y(t)} = \underset{p\times m_u \;\; m_u\times 1}{G_u^0(z)u(t)} + \underset{p\times m_d \;\; m_d\times 1}{G_d(z)d(t)} + \underset{p\times m_w \;\; m_w\times 1}{G_w(z)w(t)} + \underset{p\times m_f \;\; m_f\times 1}{G_f(z)f(t)}$$

Example: static hydraulic system

Consider a two-section pipeline system



Nominal relations between the variables are

$$u_a^0(t) = y_1^0(t) \qquad u_a^0(t) = y_2^0(t) \quad \Rightarrow \quad y_1^0(t) = y_2^0(t)$$

Example: static hydraulic system

Assume there is an **actuator fault**. Thus, the signal that enters to the system is

$$u_a^0(t) = u_a(t) + f_a$$

Assume there are two **sensors fault**. Thus, the measured signals are

$$y_1(t) = y_1^0(t) + f_{y_1} \qquad y_2(t) = y_2^0(t) + f_{y_2}$$

The nominal relations thus becomes

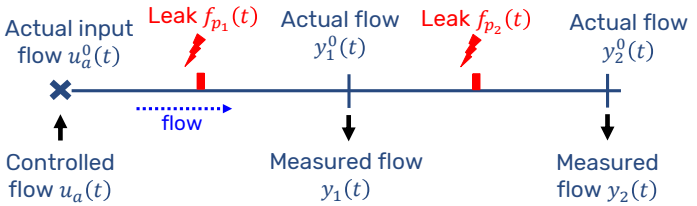
$$u_a^0(t) = y_1^0(t) \qquad u_a^0(t) = y_2^0(t)$$

$$u_a(t) + f_a = y_1(t) - f_{y_1} \qquad u_a(t) + f_a = y_2(t) - f_{y_2}$$

$$y_1(t) = u_a(t) + f_a(t) + f_{y_1}(t) \qquad y_2(t) = u_a(t) + f_a(t) + f_{y_2}(t)$$

Example: static hydraulic system

Now, assume that the system has two **leaks (process faults)**



The nominal relations become

$$y_1^0(t) = u_a^0(t) - f_{p_1}(t) \qquad y_2^0(t) = u_a^0(t) - f_{p_1}(t) - f_{p_2}(t)$$

Example: static hydraulic system

So that the relations between the measurable variables are

$$y_1(t) = u_a(t) + f_a(t) + f_{y_1}(t) - f_{p_1}(t) \qquad y_2(t) = u_a(t) + f_a(t) + f_{y_2}(t) - f_{p_1}(t) - f_{p_2}(t)$$

Thus

$$\underbrace{\begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix}}_{\boldsymbol{y}(t)} = \underbrace{\begin{bmatrix} 1 \\ 1 \end{bmatrix}}_{\boldsymbol{G}_u^0(z)} u_a(t) + \underbrace{\left[\begin{array}{c|cc|cc} 1 & -1 & 0 & 1 & 0 \\ 1 & -1 & -1 & 0 & 1 \end{array}\right]}_{\boldsymbol{G}_f(z)} \cdot \left[\begin{array}{c} f_a(t) \\ \hline f_{p_1}(t) \\ f_{p_2}(t) \\ \hline f_{y_1}(t) \\ f_{y_2}(t) \end{array}\right] \begin{matrix} \\ \\ \boldsymbol{f}_p(t) \\ \\ \boldsymbol{f}_y(t) \end{matrix}$$

$\boldsymbol{G}_a(z)\boldsymbol{S}_{f_a} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} 1$; $\boldsymbol{G}_{f_p}(z)$; $\boldsymbol{S}_{f_y}$

Multiplicative faults and disturbances

Assume a **discrepancy** between the «true» system transfer function and its model:

$$\boldsymbol{G}_u^0(z) = \boldsymbol{G}_u(z) + \boldsymbol{\Delta G}(z)$$

The discrepancies can be due to:

1. **Parametric faults**: the plant deviated from its nominal behaviour, which was properly represented by the model
2. **Modeling errors**: this may be constant and present ever since the implementation of the residual generator. It can be caused by **inaccuracies of some parameters**, or **approximation** of a higher order plant with a lower one.

   In other cases, it is due to the approximation of a **nonlinear** plant with a linear one: here, the inaccuracies depend on the operating point and so may vary with time

Multiplicative faults and disturbances

In absence of additive faults, disturbances and noises, the input-output relation becomes

$$\boldsymbol{y}^0(t) = \boldsymbol{G}_u^0(z)\boldsymbol{u}^0(t) = [\boldsymbol{G}_u(z) + \boldsymbol{\Delta G}(z)]\boldsymbol{u}^0(t) = \boldsymbol{G}_u(z)\boldsymbol{u}^0(t) + \boldsymbol{\Delta G}(z)\boldsymbol{u}^0(t)$$

Assume now that $\boldsymbol{\Delta G}(z) = \boldsymbol{\Delta G}_f(z) + \boldsymbol{\Delta G}_d(z)$, where:

- $\boldsymbol{\Delta G}_f(z)$: **parametric faults**
- $\boldsymbol{\Delta G}_d(z)$: **modelling errors**

We can write

$$\underset{p\times 1}{\boldsymbol{y}^0(t)} = \underset{p\times m_u}{\boldsymbol{G}_u(z)}\underset{m_u\times 1}{\boldsymbol{u}^0(t)} + \underset{p\times m_u}{\boldsymbol{\Delta G}_f(z)}\underset{m_u\times 1}{\boldsymbol{u}^0(t)} + \underset{p\times m_u}{\boldsymbol{\Delta G}_d(z)}\underset{m_u\times 1}{\boldsymbol{u}^0(t)}$$

Multiplicative faults and disturbances

A simplifying assumption in modelling **parametric faults** arises when considering a set of **«underlying parameters»** that characterize the plant

Consider a set of parameters $\underset{\nu\times 1}{\boldsymbol{\theta}} = [\theta_1 \;\; \theta_2 \;\; \dots \;\; \theta_\nu]^\top$ with uncertainty $\underset{\nu\times 1}{\boldsymbol{\Delta\theta}} = [\Delta\theta_1 \;\; \Delta\theta_2 \;\; \dots \;\; \Delta\theta_\nu]^\top$

Let $\boldsymbol{\theta}^*$ be the value of the parameters actually in use by the model, i.e. $\boldsymbol{G}_u(z, \boldsymbol{\theta}^*)$

The uncertainty $\boldsymbol{\Delta G}(z)$ can be approximated as

$$\underset{p\times m_u}{\boldsymbol{\Delta G}(z)} = \sum_{i=1}^{\nu} \underset{p\times m_u}{\boldsymbol{G}_{\theta,i}(z)} \cdot \underset{1\times 1}{\Delta\theta_i}$$

$$\underset{p\times m_u}{\boldsymbol{G}_{\theta,i}(z)} := \left.\frac{\partial \boldsymbol{G}_u(z,\boldsymbol{\theta})}{\partial \theta_i}\right|_{\boldsymbol{\theta}=\boldsymbol{\theta}^*} \quad i = 1, \dots, \nu$$

Advanced material

Multiplicative faults and disturbances

With this approximation, the model discrepancy term can be written as

$$\underset{p\times m_u}{\boldsymbol{\Delta G}(z)}\underset{m_u\times 1}{\boldsymbol{u}(t)} = \left[\sum_{i=1}^{\nu} \underset{p\times m_u}{\boldsymbol{G}_{\theta,i}(z)}\cdot \Delta\theta_i\right]\boldsymbol{u}(t) = \sum_{i=1}^{\nu}\left[\underset{p\times m_u}{\boldsymbol{G}_{\theta,i}(z)}\cdot \underset{m_u\times 1}{\boldsymbol{u}(t)}\right]\underset{1\times 1}{\Delta\theta_i} = \underset{p\times \nu}{\boldsymbol{N}(t)}\cdot \underset{\nu\times 1}{\boldsymbol{\Delta\theta}}$$

$$\underset{p\times \nu}{\boldsymbol{N}(t)} \coloneqq \left[\underset{p\times m_u}{\boldsymbol{G}_{\theta,1}(z)}\underset{m_u\times 1}{\boldsymbol{u}(t)} \quad \boldsymbol{G}_{\theta,2}(z)\boldsymbol{u}(t) \quad \dots \quad \boldsymbol{G}_{\theta,\nu}(z)\boldsymbol{u}(t)\right]$$

Assume that the underlying parameters vector can be decomposed as $\underset{\nu\times 1}{\boldsymbol{\theta}} = \left[\overset{\nu_f\times 1}{\boldsymbol{\theta}_f^{\mathsf{T}}} \mid \overset{\nu_d\times 1}{\boldsymbol{\theta}_d^{\mathsf{T}}}\right]^{\mathsf{T}}$ so that $\boldsymbol{\theta}_f$ represents **parametric faults** and $\boldsymbol{\theta}_d$ **modelling errors**. Then, with a proper decomposition $\underset{1\times \nu}{\boldsymbol{N}(t)} = \left[\underset{1\times \nu_f}{\boldsymbol{N}_f(t)} \mid \underset{1\times \nu_d}{\boldsymbol{N}_d(t)}\right]$, we have

$$\boldsymbol{\Delta G}_f(z)\boldsymbol{u}^0(t) + \boldsymbol{\Delta G}_d(z)\boldsymbol{u}^0(t) = \boldsymbol{N}_f(t)\boldsymbol{\Delta\theta}_f + \boldsymbol{N}_d(t)\boldsymbol{\Delta\theta}_d$$

Advanced material

Modeling of faulty systems

Considering all the external inputs, and neglecting the effect of model discrepancies on fault and disturbance matrices, we have that

$$\underset{p\times 1}{\boldsymbol{y}(t)} = \underset{p\times m_u}{\boldsymbol{G}_u(z)}\underset{m_u\times 1}{\boldsymbol{u}(t)} + \underset{p\times m_d}{\boldsymbol{G}_d(z)}\underset{m_d\times 1}{\boldsymbol{d}(t)} + \underset{p\times m_w}{\boldsymbol{G}_w(z)}\underset{m_w\times 1}{\boldsymbol{w}(t)} + \underset{p\times m_f}{\boldsymbol{G}_f(z)}\underset{m_f\times 1}{\boldsymbol{f}(t)} + \underset{p\times \nu_f}{\boldsymbol{N}_f(t)}\underset{\nu_f\times 1}{\boldsymbol{\Delta\theta}_f} + \underset{p\times \nu_d}{\boldsymbol{N}_d(t)}\underset{\nu_d\times 1}{\boldsymbol{\Delta\theta}_d}$$

In this way, model discrepancies becomes formally similar to additive faults and disturbances. However, multiplicative faults and disturbances are constant with time varying coefficients: the vice versa holds for additive faults and disturbances

	Additive	Multiplicative
Faults	sensor fault actuator fault plant fault	plant fault
Disturbances	plant disturbance	modeling error
Noises	sensor noise actuator noise plant noise	

Modeling of faulty systems

Only additive
Additive faults, disturbances, noise

$$\underset{p\times 1}{\boldsymbol{y}(t)} = \underset{p\times m_u}{\boldsymbol{G}_u^0(z)}\underset{m_u\times 1}{\boldsymbol{u}(t)} + \underset{p\times m_d}{\boldsymbol{G}_d(z)}\underset{m_d\times 1}{\boldsymbol{d}(t)} + \underset{p\times m_w}{\boldsymbol{G}_w(z)}\underset{m_w\times 1}{\boldsymbol{w}(t)} + \underset{p\times m_d}{\boldsymbol{G}_f(z)}\underset{m_d\times 1}{\boldsymbol{f}(t)}$$

Only multiplicative
Model errors, parametric faults

$$\underset{p\times 1}{\boldsymbol{y}^0(t)} = \underset{p\times m_u}{\boldsymbol{G}_u(z)}\underset{m_u\times 1}{\boldsymbol{u}^0(t)} + \underset{p\times m_u}{\boldsymbol{\Delta G}_d(z)}\underset{m_u\times 1}{\boldsymbol{u}^0(t)} + \underset{p\times m_u}{\boldsymbol{\Delta G}_f(z)}\underset{m_u\times 1}{\boldsymbol{u}^0(t)}$$

Though the **modelling discrepancy** terms appears additively, they **differ from the additive** faults and disturbances in an important way:

- **Additive faults** and **disturbances** are ***signals***, their **coefficients** are ***LTI systems***
- **Multiplicative faults** are ***LTI systems***, their **coefficients** are ***signals***

Outline

1. The model-based fault diagnosis framework
2. Modeling of faulty systems
3. **Residuals generation**

Residuals generator

Residuals are quantities that are **nominally zero**. They become nonzero in response to faults (and also disturbances, noise and modelling error)

A generic **residuals generator** is a **LTI system** fed by measured/known inputs and outputs (observables). Its generic form is:

$$\underset{q\times 1}{\boldsymbol{r}(t)} = \underset{q\times m_u}{\boldsymbol{Q}_u(z)}\underset{m_u\times 1}{\boldsymbol{u}(t)} + \underset{q\times p}{\boldsymbol{Q}_y(z)}\underset{p\times 1}{\boldsymbol{y}(t)}$$

In absence of faults, disturbances and noise, the filters $\boldsymbol{Q}_u(z)$ and $\boldsymbol{Q}_y(z)$ have to return zero residuals, i.e. when it holds that $\boldsymbol{u}(t) = \boldsymbol{u}^0(t)$, $\boldsymbol{y}(t) = \boldsymbol{y}^0(t)$, $\boldsymbol{G}_u(z) = \boldsymbol{G}_u^0(z)$ and so

$$\boldsymbol{y}(t) = \boldsymbol{G}_u(z)\boldsymbol{u}(t)$$

Residuals generator

Substituting $\boldsymbol{y}(t)$ in the residuals equation and equating to zero, the equation

$$\boldsymbol{r}(t) = \boldsymbol{Q}_u(z)\boldsymbol{u}(t) + \boldsymbol{Q}_y(z)\boldsymbol{y}(t) \;\; = \boldsymbol{Q}_u(z)\boldsymbol{u}(t) + \boldsymbol{Q}_y(z)\boldsymbol{G}_u(z)\boldsymbol{u}(t) = \boldsymbol{0}$$

has to be satisfied for every $\boldsymbol{u}(t)$, requiring that:

$$\boldsymbol{Q}_u(z) = -\boldsymbol{Q}_y(z)\boldsymbol{G}_u(z)$$

Thus, the generic residuals generator can also be written in the **computational form:**

$$\boldsymbol{r}(t) = \boldsymbol{Q}_y(z)[\boldsymbol{y}(t) - \boldsymbol{G}_u(z)\boldsymbol{u}(t)]$$

Residuals generator

Now, express $\boldsymbol{y}(t) - \boldsymbol{G}_u(z)\boldsymbol{u}(t)$ from the generic representation with external inputs:

$$\boldsymbol{y}(t) - \boldsymbol{G}_u(z)\boldsymbol{u}(t) = \boldsymbol{G}_d(z)\boldsymbol{d}(t) + \boldsymbol{G}_w(z)\boldsymbol{w}(t) + \boldsymbol{G}_f(z)\boldsymbol{f}(t) + \boldsymbol{N}_f(t)\boldsymbol{\Delta\theta}_f + \boldsymbol{N}_d(t)\boldsymbol{\Delta\theta}_d$$

The residuals generator **internal form** is expressed as:

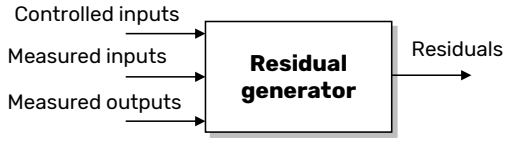
$$\boldsymbol{r}(t) = \boldsymbol{Q}_y(z)\big[\boldsymbol{G}_d(z)\boldsymbol{d}(t) + \boldsymbol{G}_w(z)\boldsymbol{w}(t) + \boldsymbol{G}_f(z)\boldsymbol{f}(t) + \boldsymbol{N}_f(t)\boldsymbol{\Delta\theta}_f + \boldsymbol{N}_d(t)\boldsymbol{\Delta\theta}_d\big]$$

The internal form shows the **effect of all external inputs** on the residuals

Residuals generator

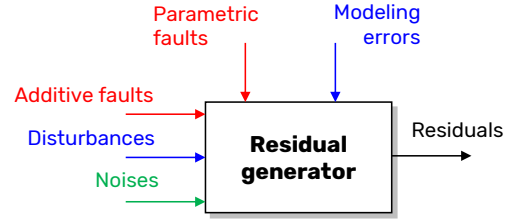
Computational form

$$\boldsymbol{r}(t) = \boldsymbol{Q}_y(z)[\boldsymbol{y}(t) - \boldsymbol{G}_u(z)\boldsymbol{u}(t)]$$



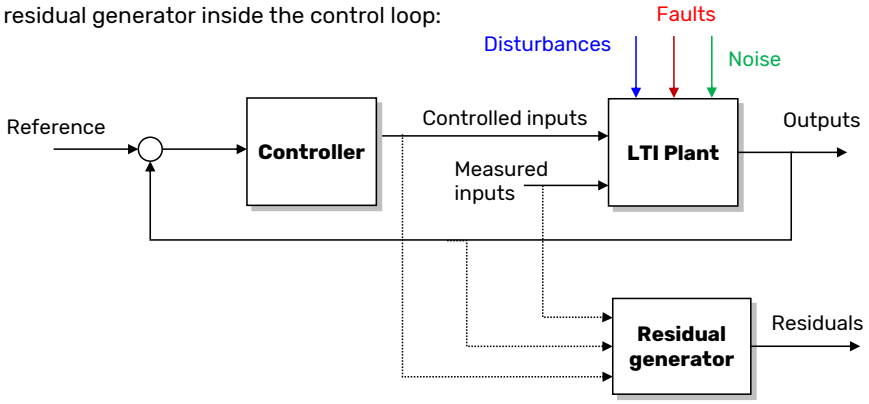
Internal form

$$\boldsymbol{r}(t) = \boldsymbol{Q}_y(z)[\boldsymbol{G}_d(z)\boldsymbol{d}(t) + \boldsymbol{G}_w(z)\boldsymbol{w}(t) + \boldsymbol{G}_f(z)\boldsymbol{f}(t) + \boldsymbol{N}_f(t)\boldsymbol{\Delta\theta}_f + \boldsymbol{N}_d(t)\boldsymbol{\Delta\theta}_d]$$

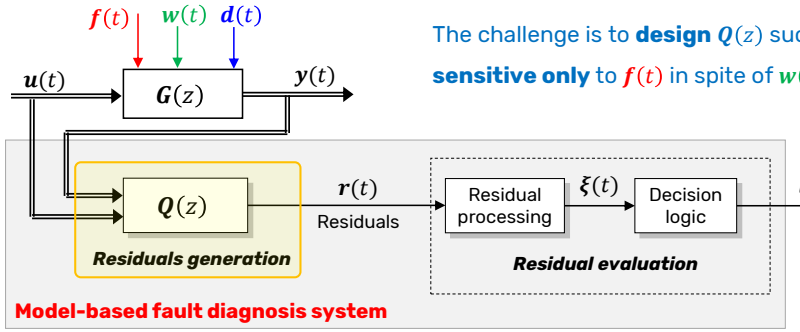


Residuals generator

The residual generator inside the control loop:



Residuals generator



The challenge is to **design** $Q(z)$ such that $r(t)$ is **sensitive only** to $f(t)$ in spite of $w(t)$ and $d(t)$

Detection properties of the residuals generator

Ideally, the **residuals** should be affected **only by the faults**. However, the presence of disturbances, noise and modelling errors also causes the residuals to become nonzero and this interferes with the detection of the faults

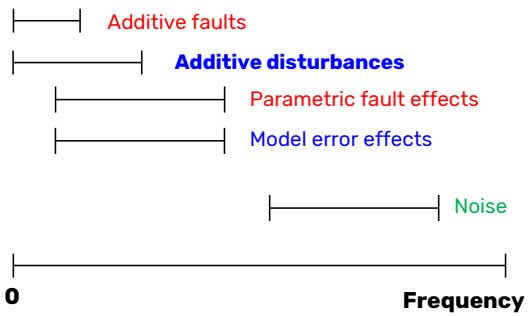
Thus, the **residuals generator** should be designed to be maximally unaffected by these nuisance inputs, i.e. to be **robust**

The treatment of the different nuisance inputs is somewhat different, primarily due to differences in their **temporal behavior** and the **prior information** on them (e.g. assumptions on the statistical distribution of the noise)

Detection properties of the residuals generator

Additive disturbances are usually «slow». Their frequency spectrum is similar to those of additive faults

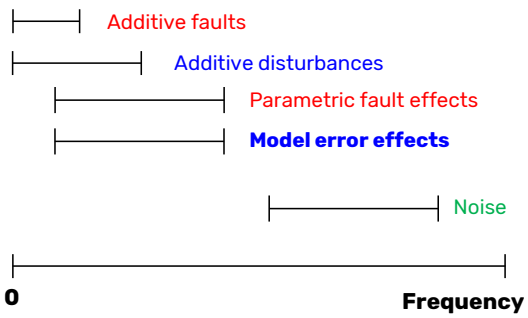
Robust residual generation can be achieved by **explicitly decoupling** the residuals from such disturbances (if there are enough output measurements)



Detection properties of the residuals generator

Multiplicative disturbances (model errors) are also usually «slow».

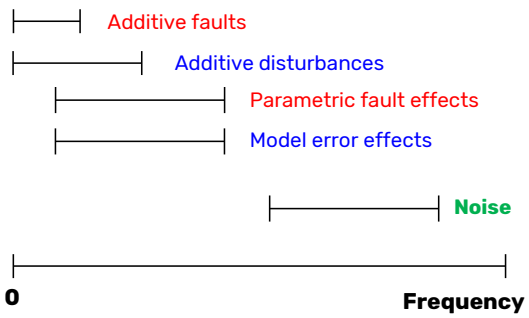
The model error is either permanently present, or arises as result of (normally slow) plant variations



Detection properties of the residuals generator

Noises are usually zero mean and at **higher frequency** range than faults and disturbances

Thus, instead of decoupling (which imposes design restrictions) noises are usually **filtered** or checked against **thresholds** (statistically determined)



Detection properties of the residual generator

A fault can be detected by comparing a **residuals evaluation** function $\xi := J(\boldsymbol{r}(t))$ with a threshold function $\eta(t)$

$$\begin{cases} \xi := J(\boldsymbol{r}(t)) \leq \eta(t) & \text{for} \quad \boldsymbol{f}(t) = 0 \\ \xi := J(\boldsymbol{r}(t)) > \eta(t) & \text{for} \quad \boldsymbol{f}(t) \neq 0 \end{cases}$$

There are many ways of defining evaluation functions and determining thresholds. As an example, the residuals evaluation function is chosen as a **norm of the residual vector** and the threshold can be chosen as a **constant positive value (fixed threshold)**

UNIVERSITÀ DEGLI STUDI DI BERGAMO | Dipartimento di Ingegneria Gestionale, dell'Informazione e della Produzione

UNIVERSITÀ DEGLI STUDI DI BERGAMO | Dipartimento di Ingegneria Gestionale, dell'Informazione e della Produzione

ADAPTIVE LEARNING, ESTIMATION AND SUPERVISION OF DYNAMICAL SYSTEMS (ALES)

Lecture 11: Parity space fault detection

Master Degree in COMPUTER ENGINEERING

Data Science and Data Engineering Curriculum

SPEAKER
Prof. Mirko Mazzoleni

PLACE
University of Bergamo

Syllabus

1. **Recursive and adaptive identification**
   - 1.1 Recursive ARX estimation (RLS)
   - 1.2 Least Mean Squares (LMS)
   - 1.3 Instrumental Variables (IV)
2. **Closed-loop identification**
3. **Subspace and MIMO identification**
   - 3.1 Singular Value Decomposition
   - 3.2 Impulse data: Ho-Kalman, Kung algorithms
   - 3.3 Generic I/O data: the MOESP algorithm
4. **Supervision of dynamical systems**
   - 4.1 Introduction to fault diagnosis
   - 4.2 Model-based fault diagnosis
   - 4.3 Parity space approaches
   - 4.4 Observer-based approaches
   - 4.5 Signal-based fault diagnosis
   - 4.6 Knowledge-based fault diagnosis

CASE STUDIES

- Virtual Reference Feedback Tuning
- Nuclear particles classification
- Leak detection in an industrial valve
- Bearing fault identification

Outline

1. The parity space framework
2. Effects of faults
3. Generic residuals generator formulation
4. Diagnostic observer

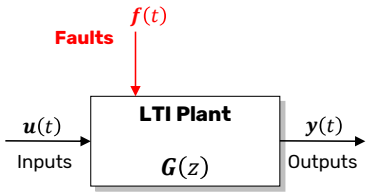
Outline

1. **The parity space framework**
2. Effects of faults
3. Generic residuals generator formulation
4. Diagnostic observer

The parity space framework

Here we present a **special case** of parity space approach known as the **Chow-Willsky** scheme, based on a **state-space** representation of the system

$$\begin{cases} \underset{n\times 1}{\boldsymbol{x}(t+1)} = \underset{n\times n}{A}\cdot\underset{n\times 1}{\boldsymbol{x}(t)} + \underset{n\times m_u}{B}\cdot\underset{m_u\times 1}{\boldsymbol{u}(t)} + \underset{n\times m_f}{B_f}\cdot\underset{m_f\times 1}{\boldsymbol{f}(t)} \\ \underset{p\times 1}{\boldsymbol{y}(t)} = \underset{p\times n}{C}\cdot\underset{n\times 1}{\boldsymbol{x}(t)} + \underset{p\times m_u}{D}\cdot\underset{m_u\times 1}{\boldsymbol{u}(t)} + \underset{p\times m_f}{D_f}\cdot\underset{m_f\times 1}{\boldsymbol{f}(t)} \end{cases}$$



- $\boldsymbol{f}(t) \in \mathbb{R}^{m_f\times 1}$: additive fault signals
- We assume that no disturbances $\boldsymbol{d}(t)$ and no noise signals $\boldsymbol{w}(t)$ are present

The parity space idea

Suppose that $\boldsymbol{w}(t) = \boldsymbol{0}$, $\boldsymbol{d}(t) = \boldsymbol{0}$ and $\boldsymbol{f}(t) = \boldsymbol{0}$, i.e. the system is **not subject** to external noise, disturbances and faults. Then, with $s \geq 0$, the following relations hold:

- $\boldsymbol{y}(t-s) = C\boldsymbol{x}(t-s) + D\boldsymbol{u}(t-s)$
- $\boldsymbol{y}(t-s+1) = C\boldsymbol{x}(t-s+1) + D\boldsymbol{u}(t-s+1)$
  $= C\cdot\{A\boldsymbol{x}(t-s) + B\boldsymbol{u}(t-s)\} + D\boldsymbol{u}(t-s+1)$
  $= CA^1\boldsymbol{x}(t-s) + CB\boldsymbol{u}(t-s) + D\boldsymbol{u}(t-s+1)$

The parity space idea

- $\boldsymbol{y}(t-s+1) = C\boldsymbol{x}(t-s+1) + D\boldsymbol{u}(t-s+1)$
  $= C\cdot\{A\boldsymbol{x}(t-s) + B\boldsymbol{u}(t-s)\} + D\boldsymbol{u}(t-s+1)$
  $= CA^1\boldsymbol{x}(t-s) + CB\boldsymbol{u}(t-s) + D\boldsymbol{u}(t-s+1)$
- $\boldsymbol{y}(t-s+2) = C\boldsymbol{x}(t-s+2) + D\boldsymbol{u}(t-s+2)$
  $= C\cdot\{A\boldsymbol{x}(t-s+1) + B\boldsymbol{u}(t-s+1)\} + D\boldsymbol{u}(t-s+2)$
  $= CA\cdot\{A\boldsymbol{x}(t-s) + B\boldsymbol{u}(t-s)\} + CB\boldsymbol{u}(t-s+1) + D\boldsymbol{u}(t-s+2)$
  $= CA^2\boldsymbol{x}(t-s) + CA^{2-1}B\boldsymbol{u}(t-s) + CB\boldsymbol{u}(t-s+1) + D\boldsymbol{u}(t-s+2)$
  $t-s+(2-1)$ $t-s+(2)$

The parity space idea

- $\boldsymbol{y}(t-s+2) = C\boldsymbol{x}(t-s+2) + D\boldsymbol{u}(t-s+2)$
  $= C\cdot\{A\boldsymbol{x}(t-s+1) + B\boldsymbol{u}(t-s+1)\} + D\boldsymbol{u}(t-s+2)$
  $= CA\cdot\{A\boldsymbol{x}(t-s) + B\boldsymbol{u}(t-s)\} + CB\boldsymbol{u}(t-s+1) + D\boldsymbol{u}(t-s+2)$
  $= CA^2\boldsymbol{x}(t-s) + CA^{2-1}B\boldsymbol{u}(t-s) + CB\boldsymbol{u}(t-s+1) + D\boldsymbol{u}(t-s+2)$
  $t-s+(2-1)$ $t-s+(2)$

⋮

- $\boldsymbol{y}(t-s+s) = \boldsymbol{y}(t) = CA^s\boldsymbol{x}(t-s) + CA^{s-1}B\boldsymbol{u}(t-s) + \cdots + CB\boldsymbol{u}(t-1) + D\boldsymbol{u}(t)$
  $t-s+(s-1)$ $t-s+(s)$

The parity space idea

Define the following quantities (notice the similarity with the subspace approach in Lecture 06):

$$\underset{m_u(s+1)\times 1}{\overleftarrow{\boldsymbol{u}_{s+1}}}(t) = \begin{bmatrix} \underset{m_u\times 1}{\boldsymbol{u}(t-s)} \\ \boldsymbol{u}(t-s+1) \\ \vdots \\ \boldsymbol{u}(t) \end{bmatrix} \qquad \underset{p(s+1)\times 1}{\overleftarrow{\boldsymbol{y}_{s+1}}}(t) = \begin{bmatrix} \underset{p\times 1}{\boldsymbol{y}(t-s)} \\ \boldsymbol{y}(t-s+1) \\ \vdots \\ \boldsymbol{y}(t) \end{bmatrix}$$

$$\underset{p(s+1)\times m_u(s+1)}{\mathcal{T}_{s+1}} = \begin{bmatrix} \overset{p\times m_u}{D} & 0 & \cdots & \cdots & 0 \\ CB & \ddots & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ CA^{s-1}B & \cdots & \cdots & CB & D \end{bmatrix} \qquad \underset{p(s+1)\times n}{\mathcal{O}_{s+1}} = \begin{bmatrix} \overset{p\times n}{C} \\ CA \\ \vdots \\ CA^s \end{bmatrix}$$

The parity space idea

Given the quantities $\overleftarrow{\boldsymbol{u}_{s+1}}$, $\overleftarrow{\boldsymbol{y}_{s+1}}$, $\mathcal{T}_{s+1}$, $\mathcal{O}_{s+1}$ we can write

$$\overleftarrow{\boldsymbol{y}_{s+1}}(t) = \mathcal{O}_{s+1}\cdot\boldsymbol{x}(t-s) + \mathcal{T}_{s+1}\cdot\overleftarrow{\boldsymbol{u}_{s+1}}(t)$$

Parity relations

$$\underset{p(s+1)\times 1}{\begin{bmatrix} \boldsymbol{y}(t-s) \\ \boldsymbol{y}(t-s+1) \\ \vdots \\ \boldsymbol{y}(t) \end{bmatrix}} = \underset{p(s+1)\times n}{\begin{bmatrix} C \\ CA \\ \vdots \\ CA^s \end{bmatrix}}\cdot\underset{n\times 1}{\boldsymbol{x}(t-s)} + \underset{p(s+1)\times m_u(s+1)}{\begin{bmatrix} D & 0 & \cdots & \cdots & 0 \\ CB & \ddots & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ CA^{s-1}B & \cdots & \cdots & CB & D \end{bmatrix}}\cdot\underset{m_u(s+1)\times 1}{\begin{bmatrix} \boldsymbol{u}(t-s) \\ \boldsymbol{u}(t-s+1) \\ \vdots \\ \boldsymbol{u}(t) \end{bmatrix}}$$

The parity space idea

$$\overleftarrow{\boldsymbol{y}_{s+1}}(t) = \mathcal{O}_{s+1}\cdot\boldsymbol{x}(t-s) + \mathcal{T}_{s+1}\cdot\overleftarrow{\boldsymbol{u}_{s+1}}(t)$$

- These equations constitutes a **temporal redundancy** between the variables of the system
- $\overleftarrow{\boldsymbol{y}_{s+1}}(t)$ and $\overleftarrow{\boldsymbol{u}_{s+1}}(t)$ consist of temporal past outputs and inputs, and are **known**
- Matrices $\mathcal{O}_{s+1}$ and $\mathcal{T}_{s+1}$ are composed of system matrices A, B, C, D and are also **known**
- The only **unknown** variable is $\boldsymbol{x}(t-s)$

The parity space idea

Assume that $s+1 > n$. From the Cayley-Hamilton theorem, it holds that

$$\underset{p(s+1)\times n}{\operatorname{rank}(\mathcal{O}_{s+1})} \le n < \text{number of rows of the matrix } \mathcal{O}_{s+1}$$

Then, matrix $\mathcal{O}_{s+1}$ has a **left nullspace** $\mathcal{N}(\mathcal{O}_{s+1}^{\top})$ of dimensions $q := p(s+1) - \operatorname{rank}(\mathcal{O}_{s+1}) > 0$

Therefore, there **exists** a $q \times p(s+1)$ matrix V_{s+1}, with rows $\boldsymbol{v}_{s+1}^{\top} \in \mathcal{N}(\mathcal{O}_{s+1}^{\top})$, $\boldsymbol{v}_{s+1} \neq \boldsymbol{0}$, such that:

$$\underset{q\times p(s+1)}{V_{s+1}}\cdot\underset{p(s+1)\times n}{\mathcal{O}_{s+1}} = \underset{q\times n}{\boldsymbol{0}}$$

The parity space idea

From the observation that $V_{s+1}\mathcal{O}_{s+1} = \mathbf{0}$, a **residuals signal** $\boldsymbol{r}(t) \in \mathbb{R}^{q\times 1}$ is built as:

$$\underset{q\times 1}{\boldsymbol{r}(t)} = \underset{q\times p(s+1)}{V_{s+1}} \cdot \big(\underset{p(s+1)\times 1}{\overline{\boldsymbol{y}_{s+1}}(t)} - \underset{p(s+1)\times m_u(s+1)}{\mathcal{T}_{s+1}} \cdot \underset{m_u(s+1)\times 1}{\overline{\boldsymbol{u}_{s+1}}(t)}\big)$$

Residuals generator (computational form)

In the nominal case where $\boldsymbol{d}(t) = \mathbf{0}, \boldsymbol{f}(t) = \mathbf{0}$, we have that:

$$\boldsymbol{r}(t) = V_{s+1} \cdot \big(\mathcal{O}_{s+1} \cdot \boldsymbol{x}(t-s)\big) = \mathbf{0}$$

In the nominal case, the residual is zero

Vectors $\boldsymbol{v}_{s+1}^{\mathsf{T}}$ satisfying $V_{s+1} \cdot \mathcal{O}_{s+1} = \mathbf{0}$ are called **parity vectors**.

The **left nullspace** $\mathcal{N}(\mathcal{O}_{s+1}^{\mathsf{T}})$ is called **parity space**

The parity space idea

Remark

The condition $s+1 > n$ is **only sufficient** for the existence of a left nullspace with dimension greater than zero

The **necessary (and sufficient)** condition is that

$$p(s+1) - \operatorname{rank}(\mathcal{O}_{s+1}) > 0$$

So, if the system has $p > 1$ **independent outputs**, it is possible for s to be less than n and still satisfy the parity condition

Example: parity space for SISO systems

Consider the SISO nominal system model

$$y(t) = G_u^0(z)u(t) = \frac{b_n z^n + b_{n-1}z^{n-1} + \cdots + b_1 z + b_0}{z^n + a_{n-1}z^{n-1} + \cdots + a_1 z + a_0}u(t)$$

A trivial way to construct a parity space based residual generator is to:

1. rewrite the system into its minimum state space realization form
2. find vectors $\boldsymbol{v}_{s+1}^{\mathsf{T}} \in \mathcal{N}(\mathcal{O}_{s+1}^{\mathsf{T}})$ so that $V_{s+1} \cdot \mathcal{O}_{s+1} = \mathbf{0}$
3. construct the residual generator $\boldsymbol{r}(t) = V_{s+1} \cdot \big(\overline{\boldsymbol{y}_{s+1}}(t) - \mathcal{T}_{s+1} \cdot \overline{\boldsymbol{u}_{s+1}}(t)\big)$

Example: parity space for SISO systems

On the other side, it follows from **Cayley–Hamilton theorem** that:

$$A^n + a_{n-1}A^{n-1} + a_{n-2}A^{n-2} + \cdots + a_0 I_n = \mathbf{0} \quad \Rightarrow \quad [a_0 \quad \cdots \quad a_{n-1} \quad 1]\begin{bmatrix} C \\ CA \\ \vdots \\ CA^n \end{bmatrix} = \mathbf{0}$$

where A, C denote the system matrices of the minimum state space realization of $G_u^0(z)$

That means that:

$$\boldsymbol{v}_{s+1}^{\mathsf{T}} = [a_0 \quad \cdots \quad a_{n-1} \quad 1]$$

is a parity space vector for $G_u^0(z)$, with $s+1 = n+1$, i.e. $s = n$. It follows that $q = p(s+1) - \operatorname{rank}(\mathcal{O}_{s+1}) = 1(n+1) - n = 1$ so that we have only a single vector $\boldsymbol{v}_{s+1}^{\mathsf{T}}$

Example: parity space for SISO systems

The residual generator can be constructed as:

$$r(t) = \boldsymbol{v}_{s+1}^{\top} \cdot \left(\overleftarrow{\boldsymbol{y}_{s+1}}(t) - \mathcal{T}_{s+1} \cdot \overleftarrow{\boldsymbol{u}_{s+1}}(t)\right) = \boldsymbol{v}_{s+1}^{\top} \overleftarrow{\boldsymbol{y}_{s+1}}(t) - \boldsymbol{v}_{s+1}^{\top} \mathcal{T}_{s+1} \cdot \overleftarrow{\boldsymbol{u}_{s+1}}(t)$$

$$= [a_0 \quad \cdots \quad a_{n-1} \quad 1] \overleftarrow{\boldsymbol{y}_{s+1}}(t) - \boldsymbol{v}_{s+1}^{\top} \mathcal{T}_{s+1} \cdot \overleftarrow{\boldsymbol{u}_{s+1}}(t)$$

$$= a_0 y(t-s) + \cdots + a_{n-1} y(t-1) + y(t) - \boldsymbol{v}_{s+1}^{\top} \mathcal{T}_{s+1} \cdot \overleftarrow{\boldsymbol{u}_{s+1}}(t) = 0$$

For the equality $r(t) = 0$ to hold in the nominal case, it follows that

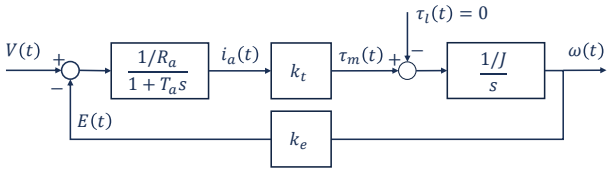
$$\boldsymbol{v}_{s+1}^{\top} \mathcal{T}_{s+1} = [b_0 \quad \cdots \quad b_{n-1} \quad b_n]$$

As a result, the residual generator, corresponding to the previous choice of $\boldsymbol{v}_s$, is given by

$$r(t) = [a_0 \quad \cdots \quad a_{n-1} \quad 1] \cdot \overleftarrow{\boldsymbol{y}_{s+1}}(t) - [b_0 \quad \cdots \quad b_{n-1} \quad b_n] \cdot \overleftarrow{\boldsymbol{u}_{s+1}}(t)$$

Example: parity space for a DC motor

Consider a **continuous-time** open-loop model of a DC motor



- Model input $u(t) = V(t)$
- Model output $y(t) = \omega(t)$
- Disturbance $d(t) = \tau_l(t)$

- Total inertia J: $80.45 \cdot 10^{-6}$ Kg · m^2
- Motor electrical constant k_e: $6.27 \cdot 10^{-3}$ V/rpm
- Motor torque constant $k_t = 0.06$ Nm/A
- Motor coil inductance $L_a = 0.003$ H
- Motor coil resistance $R_a = 3.13\ \Omega$
- Electrical time constant $T_a = L_a/R_a$ s

$$\Rightarrow \quad G_u^0(s) = \frac{1}{k_e \cdot \left(1 + J \cdot \frac{R_a}{k_t k_e} s + J \cdot T_a \cdot \frac{R_a}{k_t k_e} s^2\right)}$$

Example: parity space for a DC motor

Compute the transfer function and convert to **discrete time** (with sampling freq. 100 Hz)

$$G_u^0(s) = \frac{2.486 \cdot 10^5}{(s+1042)(s+1.496)} \quad \xrightarrow{\texttt{c2d}} \quad G_u^0(z) = \frac{1.184\, z + 1.184}{z^2 - 0.9852\, z + 2.943 \cdot 10^{-5}}$$

A **valid parity vector** is:

$$\boldsymbol{v}_{s+1}^{\top} = [2.943 \cdot 10^{-5} \quad -0.9852 \quad 1]$$

Example: parity space for a DC motor

A state space realization of $G_u^0(z)$ leads to

$$G_u^0(z) = \frac{1.184\, z + 1.184}{z^2 - 0.9852\, z + 2.943 \cdot 10^{-5}} \quad \xrightarrow{\texttt{minreal(ss(G\_yu))}}$$

$$\underset{n \times n \atop 2 \times 2}{A} = \begin{bmatrix} 0.9852 & -0.007535 \\ 0.003906 & 0 \end{bmatrix} \qquad \underset{n \times m_u \atop 2 \times 1}{B} = \begin{bmatrix} 16 \\ 0 \end{bmatrix}$$

$$\underset{p \times n \atop 1 \times 2}{C} = [0.07401 \quad 18.95] \qquad \underset{p \times m_u \atop 1 \times 1}{D} = 0$$

$$\underset{p(s+1) \times n \atop 1(2+1) \times 2}{\mathcal{O}_{s+1}} = \begin{bmatrix} 0.0740 & 18.9468 \\ 0.1469 & -0.0006 \\ 0.1447 & -0.0011 \end{bmatrix} \qquad \underset{p(s+1) \times m_u(s+1) \atop 1(2+1) \times 1(2+1)}{\mathcal{T}_{s+1}} = \begin{bmatrix} 0 & 0 & 0 \\ 1.1842 & 0 & 0 \\ 2.3508 & 1.1842 & 0 \end{bmatrix}$$

Always check the solution!

$$\boldsymbol{v}_{s+1}^{\top} \mathcal{O}_{s+1} = [2.943 \cdot 10^{-5} \quad -0.9852 \quad 1] \cdot \begin{bmatrix} 0.0740 & 18.9468 \\ 0.1469 & -0.0006 \\ 0.1447 & -0.0011 \end{bmatrix} \approx [0 \quad 0]$$ **CORRECT!** ☺

Example: parity space for a DC motor

1. Simulate data

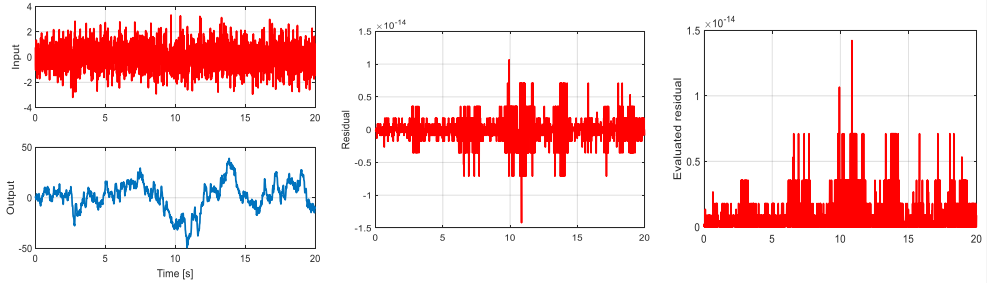
$u(t) = \mathrm{WN}(0,1)$

2. Compute residual

$r(t) = \boldsymbol{v}_{s+1}^{\top} \cdot \left(\overleftarrow{\boldsymbol{y}_{s+1}}(t) - \mathcal{T}_{s+1} \cdot \overleftarrow{\boldsymbol{u}_{s+1}}(t)\right)$

3. Evaluate residual

$\xi(t) = |r(t)|$



Outline

1. The parity space framework
2. **Effects of faults**
3. Generic residuals generator formulation
4. Diagnostic observer

Parity space: effects of faults

Suppose now that some **faults** affect the system, i.e. $\boldsymbol{f}(t) \neq 0$. Define the quantities:

$$\underset{m_f(s+1)\times 1}{\overleftarrow{\boldsymbol{f}_{s+1}}}(t) = \begin{bmatrix} \underset{m_f\times 1}{\boldsymbol{f}(t-s)} \\ \boldsymbol{f}(t-s+1) \\ \vdots \\ \boldsymbol{f}(t) \end{bmatrix} \qquad \underset{p(s+1)\times m_f(s+1)}{\mathcal{F}_{s+1}} = \begin{bmatrix} D_f & 0 & \cdots & \cdots & 0 \\ \underset{p\times m_f}{CB_f} & \ddots & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ CA^{s-1}B_f & \cdots & \cdots & CB_f & D_f \end{bmatrix}$$

Then, the parity relations become

$$\overleftarrow{\boldsymbol{y}_{s+1}}(t) = \mathcal{O}_{s+1} \cdot \boldsymbol{x}(t-s) + \mathcal{T}_{s+1} \cdot \overleftarrow{\boldsymbol{u}_{s+1}}(t) + \mathcal{F}_{s+1} \cdot \overleftarrow{\boldsymbol{f}_{s+1}}(t)$$

Parity space: effects of faults and disturbances

The **effects** of the **faults** on the **residuals** are given by

$$\boldsymbol{r}(t) = V_{s+1} \cdot \left(\overleftarrow{\boldsymbol{y}_{s+1}}(t) - \mathcal{T}_{s+1} \cdot \overleftarrow{\boldsymbol{u}_{s+1}}(t)\right) = V_{s+1} \cdot \left(\mathcal{O}_{s+1} \cdot \boldsymbol{x}(t-s) + \mathcal{F}_{s+1} \cdot \overleftarrow{\boldsymbol{f}_{s+1}}(t)\right)$$

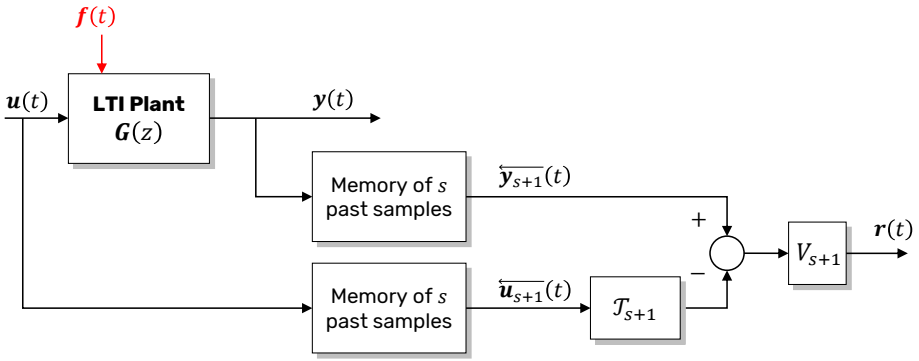
$$\underset{q\times 1}{\boldsymbol{r}(t)} = \underset{q\times p(s+1)}{V_{s+1}} \cdot \left(\underset{p(s+1)\times m_f(s+1)}{\mathcal{F}_{s+1}} \cdot \underset{m_f(s+1)\times 1}{\overleftarrow{\boldsymbol{f}_{s+1}}}(t)\right)$$

Residuals generator (internal form)

If it holds that $V_{s+1}\mathcal{F}_{s+1} \neq \mathbf{0}$, the residual is **not zero** when a fault is present, and so **fault detection is achieved**. Notice that we do not have specified any specific response from faults to residuals

Parity space: effects of faults and disturbances

A schematic of the parity space residuals generator is thus as follows



Outline

Parity space: equivalent generic formulation

The **parity space** residuals generator can be expressed in the **generic** residuals generator form

$$\boldsymbol{r}(t) = \boldsymbol{Q}_y(z)[\boldsymbol{y}(t) - \boldsymbol{G}_u(z)\boldsymbol{u}(t)]$$

Consider fo simplicity a single parity vector

$$\underset{1\times p(s+1)}{\boldsymbol{v}_{s+1}^\top} = \begin{bmatrix} \underset{1\times p}{\boldsymbol{v}_{(s)}^\top} & \underset{1\times p}{\boldsymbol{v}_{(s-1)}^\top} & \cdots & \underset{1\times p}{\boldsymbol{v}_{(0)}^\top} \end{bmatrix}$$

Define

$$\underset{1\times m_u(s+1)}{\boldsymbol{\tau}_{s+1}^\top} \coloneqq -\underset{1\times p(s+1)}{\boldsymbol{v}_{s+1}^\top} \cdot \underset{p(s+1)\times m_u(s+1)}{\mathcal{T}_{s+1}} = \begin{bmatrix} \underset{1\times m_u}{\boldsymbol{\tau}_{(s)}^\top} & \underset{1\times m_u}{\boldsymbol{\tau}_{(s-1)}^\top} & \cdots & \underset{1\times m_u}{\boldsymbol{\tau}_{(0)}^\top} \end{bmatrix}$$

$$\underset{1\times m_f(s+1)}{\boldsymbol{\zeta}_{s+1}^\top} \coloneqq \underset{1\times p(s+1)}{\boldsymbol{v}_{s+1}^\top} \cdot \underset{p(s+1)\times m_f(s+1)}{\mathcal{F}_{s+1}} = \begin{bmatrix} \underset{1\times m_f}{\boldsymbol{\zeta}_{(s)}^\top} & \underset{1\times m_f}{\boldsymbol{\zeta}_{(s-1)}^\top} & \cdots & \underset{1\times m_f}{\boldsymbol{\zeta}_{(0)}^\top} \end{bmatrix}$$

Advanced material

Parity space: equivalent generic formulation

Introduce the polynomial vectors

$$\underset{1\times p}{\boldsymbol{v}^\top(z)} \coloneqq \underset{1\times p}{\boldsymbol{v}_{(0)}^\top} + \underset{1\times p}{\boldsymbol{v}_{(1)}^\top} z^{-1} + \cdots + \underset{1\times p}{\boldsymbol{v}_{(s)}^\top} z^{-s}$$

$$\underset{1\times m_u}{\boldsymbol{\tau}^\top(z)} \coloneqq \underset{1\times m_u}{\boldsymbol{\tau}_{(0)}^\top} + \underset{1\times m_u}{\boldsymbol{\tau}_{(1)}^\top} z^{-1} + \cdots + \underset{1\times m_u}{\boldsymbol{\tau}_{(s)}^\top} z^{-s}$$

$$\underset{1\times m_f}{\boldsymbol{\zeta}^\top(z)} \coloneqq \underset{1\times m_f}{\boldsymbol{\zeta}_{(0)}^\top} + \underset{1\times m_f}{\boldsymbol{\zeta}_{(1)}^\top} z^{-1} + \cdots + \underset{1\times m_f}{\boldsymbol{\zeta}_{(s)}^\top} z^{-s}$$

We can thus write the parity space residual

$$r(t) = \boldsymbol{v}_{s+1}^\top \cdot \left(\overleftarrow{\boldsymbol{y}_{s+1}}(t) - \mathcal{T}_{s+1} \cdot \overleftarrow{\boldsymbol{u}_{s+1}}(t)\right) = \boldsymbol{v}_{s+1}^\top \cdot \mathcal{F}_{s+1} \cdot \overleftarrow{\boldsymbol{f}_{s+1}}(t)$$

as

$$r(t) = \boldsymbol{v}^\top(z) \cdot \boldsymbol{y}(t) + \boldsymbol{\tau}^\top(z) \cdot \boldsymbol{u}(t) = \boldsymbol{\zeta}^\top(z) \cdot \boldsymbol{f}(t)$$

Advanced material

Parity space: equivalent generic formulation

In the fault-free nominal case, it holds that

$$r(t) = \boldsymbol{v}^\top(z) \cdot \boldsymbol{y}(t) + \boldsymbol{\tau}^\top(z) \cdot \boldsymbol{u}(t) = 0$$

$$= \boldsymbol{v}^\top(z) \cdot \boldsymbol{G}_u(z)\boldsymbol{u}(t) + \boldsymbol{\tau}^\top(z) \cdot \boldsymbol{u}(t) = 0$$

so that

$$\boldsymbol{\tau}^\top(z) \cdot \boldsymbol{u}(t) = -\boldsymbol{v}^\top(z) \cdot \boldsymbol{G}_u(z)\boldsymbol{u}(t)$$

Therefore, considering again the faults

$$r(t) = \boldsymbol{v}^\top(z) \cdot \boldsymbol{y}(t) + \boldsymbol{\tau}^\top(z) \cdot \boldsymbol{u}(t) = \boldsymbol{\zeta}^\top(z) \cdot \boldsymbol{f}(t)$$

$$= \boldsymbol{v}^\top(z) \cdot \big(\boldsymbol{y}(t) - \boldsymbol{G}_u(z)\boldsymbol{u}(t)\big) = \boldsymbol{\zeta}^\top(z) \cdot \boldsymbol{f}(t)$$

Parity space: equivalent generic formulation

Considering a generic number of q residuals, we can express the computational form as

$$\underset{q\times 1}{\boldsymbol{r}(t)} = \underset{q\times p}{V(z)} \cdot \big(\boldsymbol{y}(t) - \boldsymbol{G}_u(z)\boldsymbol{u}(t)\big)$$

where $V(z)$ stacks the q polynomials $\boldsymbol{v}^\top(z)$. This form is equivalent to the general one by setting $\boldsymbol{Q}_y(z) = V(z)$

Example: parity space for a MIMO system

Consider the following MIMO system with $m_u = 2, p = 3, m_f = 1$, sampled at $T_s = 1$ s

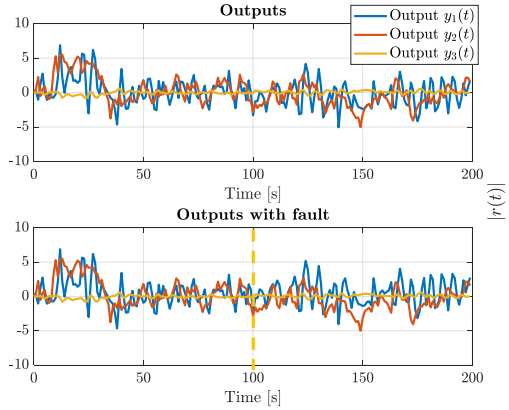
$$\underset{n\times n\ 4\times 4}{A} = \begin{bmatrix} 0.5 & -0.7 & 0.7 & 0 \\ 0 & 0.8 & 0.06 & 0 \\ -1 & 0 & 0 & 0.1 \\ 0 & 0 & -0.1 & 0.4 \end{bmatrix} \quad \underset{n\times m_u\ 4\times 2}{B} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \quad \underset{p\times n\ 3\times 4}{C} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \underset{p\times m_u\ 3\times 2}{D} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

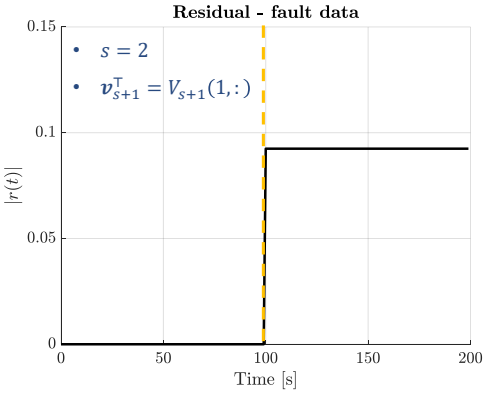
$$\underset{n\times m_f\ 2\times 1}{B_f} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \underset{n\times m_f\ 2\times 1}{D_f} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

$$\underset{p\times n}{\boldsymbol{y}(t)} = \underset{p\times m_u}{\boldsymbol{G}_u(z)}\underset{m_u\times 1}{\boldsymbol{u}(t)} + \underset{p\times m_f}{\boldsymbol{G}_f(z)}\underset{m_f\times 1}{\boldsymbol{f}(t)}$$

Collect $N = 200$ data. The inputs are white noises with zero mean and variance 1. A step-like fault of amplitude 1 acts on the system at time $t = 100$ s.

Example: parity space for a MIMO system





Outline

1. The parity space framework
2. Effects of faults
3. Generic residuals generator formulation
4. **Diagnostic observer**

Diagnostic observer

A **disadvantage** of the parity approach method is that it requires to **store a sample** of past inputs and outputs at times $t-1, \dots, t-s$, to compute the residual signal at the current time t. The implementation is **non-recursive**

It is possible to employ a different residual generator scheme, known as **diagnostic observer**, by using a designed parity vector $\boldsymbol{v}_{s+1}$

This allows to employ a **recursive implementation**, s.t. $r(t)$ is computed from $r(t-1)$

A common strategy based on **«parity space design, observer-based implementation»** is vastly used. We present a diagnostic observer for a scalar residual

Advanced material

Diagnostic observer

A **diagnostic observer** can be formulated as the following **dynamical system**

$$\begin{cases} \underset{s\times 1}{\boldsymbol{m}(t+1)} = \underset{s\times s}{F}\cdot\underset{s\times 1}{\boldsymbol{m}(t)} + \underset{s\times p}{K}\cdot\underset{p\times 1}{\boldsymbol{y}(t)} + \underset{s\times m_u}{J}\cdot\underset{m_u\times 1}{\boldsymbol{u}(t)} \\ \underset{1\times 1}{r(t)} = \underset{1\times s}{L_1}\cdot\underset{s\times 1}{\boldsymbol{m}(t)} + \underset{1\times p}{L_2}\cdot\underset{p\times 1}{\boldsymbol{y}(t)} + \underset{1\times m_u}{L_3}\cdot\underset{m_u\times 1}{\boldsymbol{u}(t)} \end{cases}$$

We can interpret $\hat{\boldsymbol{o}}(t) \coloneqq -L_1\cdot\boldsymbol{m}(t) - L_3\cdot\boldsymbol{u}(t)$ as an **estimate** for $\boldsymbol{o} \coloneqq L_2\cdot\boldsymbol{y}(t)$. In this way, the residual signal reads as $r(t) = \boldsymbol{o}(t) - \hat{\boldsymbol{o}}(t)$

Notice how $\hat{\boldsymbol{o}}(t)$ depends on $\boldsymbol{m}(t)$, which in turn depends on $\boldsymbol{y}(t-1)$

Advanced material

Diagnostic observer

Let $\underset{1\times p}{\boldsymbol{v}_{s+1}^\top} = \begin{bmatrix}\boldsymbol{v}_{(s)}^\top & \boldsymbol{v}_{(s-1)}^\top & \cdots & \boldsymbol{v}_{(0)}^\top\end{bmatrix}$, be a parity vector. Then, we have:

$$\underset{s\times(s-1)}{F} = [F_0 \quad \boldsymbol{k}] \qquad F_0 = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ 1 & 0 & 0 & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 1 & 0 \\ 0 & \cdots & 0 & 1 \end{bmatrix} \qquad \boldsymbol{k} = \begin{bmatrix} k_s \\ k_{s-1} \\ \vdots \\ k_1 \end{bmatrix} \qquad \underset{s\times p}{K} = -\underset{s\times p}{\begin{bmatrix}\boldsymbol{v}_{(s)} \\ \vdots \\ \boldsymbol{v}_{(1)}\end{bmatrix}} - \underset{s\times 1}{\boldsymbol{k}}\cdot\underset{1\times p}{\boldsymbol{v}_{(0)}}$$

$$\underset{(s+1)\times m_u}{\begin{bmatrix} J \\ -L_3 \end{bmatrix}} = \underbrace{\begin{bmatrix} \boldsymbol{v}_{(s)} + k_s\boldsymbol{v}_{(0)} & \boldsymbol{v}_{(s-1)} & \boldsymbol{v}_{(s-2)} & \cdots & \boldsymbol{v}_{(1)} & \boldsymbol{v}_{(0)} \\ \boldsymbol{v}_{(s-1)} + k_{s-1}\boldsymbol{v}_{(0)} & \boldsymbol{v}_{(s-2)} & \cdots & \cdots & \boldsymbol{v}_{(0)} & \boldsymbol{0} \\ \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\ \boldsymbol{v}_{(1)} + k_1\boldsymbol{v}_{(0)} & \boldsymbol{v}_{(0)} & \boldsymbol{0} & \cdots & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{v}_{(0)} & \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{0} & \boldsymbol{0} \end{bmatrix}}_{1\times p(s+1)} \underbrace{\begin{bmatrix} D \\ CB \\ CAB \\ \vdots \\ CA^{s-2}B \\ CA^{s-1}B \end{bmatrix}}_{p(s+1)\times m_u}$$

$s\times p(s+1)$

$$\underset{1\times s}{L_1} = -[0 \quad \cdots \quad 1] \qquad \underset{1\times p}{L_2} = \boldsymbol{v}_{(0)}$$

Advanced material

Diagnostic observer

$$\begin{cases} \underset{s\times 1}{\boldsymbol{m}(t+1)} = \underset{s\times s}{F}\cdot\underset{s\times 1}{\boldsymbol{m}(t)} + \underset{s\times p}{K}\cdot\underset{p\times 1}{\boldsymbol{y}(t)} + \underset{s\times m_u}{J}\cdot\underset{m_u\times 1}{\boldsymbol{u}(t)} \\ \underset{1\times 1}{r(t)} = \underset{1\times s}{L_1}\cdot\underset{s\times 1}{\boldsymbol{m}(t)} + \underset{1\times p}{L_2}\cdot\underset{p\times 1}{\boldsymbol{y}(t)} + \underset{1\times m_u}{L_3}\cdot\underset{m_u\times 1}{\boldsymbol{u}(t)} \end{cases}$$

The vector $\boldsymbol{k}$ in the matrix F is an **additional degree of freedom** given by the observer formulation. It can be used to assign a **desired dynamic** to the observer

When $\boldsymbol{k} = \boldsymbol{0}$, then the DO formulation is **analogous** to the parity space design

The obtained residual generator is a **dynamical system** (a **filter)**, that takes as input the **system inputs** $u(t)$ and **outputs** $y(t)$, and get as output the scalar **residual signal** $r(t)$

UNIVERSITÀ DEGLI STUDI DI BERGAMO Dipartimento di Ingegneria Gestionale, dell'Informazione e della Produzione

UNIVERSITÀ DEGLI STUDI DI BERGAMO Dipartimento di Ingegneria Gestionale, dell'Informazione e della Produzione

ADAPTIVE LEARNING, ESTIMATION AND SUPERVISION OF DYNAMICAL SYSTEMS (ALES)

Lecture 12: Observers fault detection

Master Degree in COMPUTER ENGINEERING

Data Science and Data Engineering Curriculum

SPEAKER
Prof. Mirko Mazzoleni

PLACE
University of Bergamo

Syllabus

1. **Recursive and adaptive identification**
   - 1.1 Recursive ARX estimation (RLS)
   - 1.2 Least Mean Squares (LMS)
   - 1.3 Instrumental Variables (IV)
2. **Closed-loop identification**
3. **Subspace and MIMO identification**
   - 3.1 Singular Value Decomposition
   - 3.2 Impulse data: Ho-Kalman, Kung algorithms
   - 3.3 Generic I/O data: the MOESP algorithm
4. **Supervision of dynamical systems**
   - 4.1 Introduction to fault diagnosis
   - 4.2 Model-based fault diagnosis
   - 4.3 Parity space approaches
   - 4.4 Observer-based approaches
   - 4.5 Signal-based fault diagnosis
   - 4.6 Knowledge-based fault diagnosis

CASE STUDIES

- Virtual Reference Feedback Tuning
- Nuclear particles classification
- Leak detection in an industrial valve
- Bearing fault identification

Outline

1. The observer-based framework
2. Deterministic setting: functional observers
3. Functional (output) observers for residual generation
4. Stochastic setting: Kalman-filter

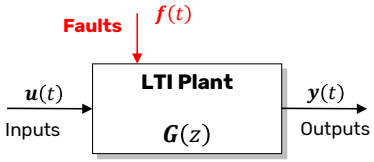
Outline

1. **The observer-based framework**
2. Deterministic setting: functional observers
3. Functional (output) observers for residual generation
4. Stochastic setting: Kalman-filter

The observer-based framework to fault diagnosis

Consider a LTI dynamical system with **faults**

$$\begin{cases} \underset{n\times 1}{\boldsymbol{x}(t+1)} = \underset{n\times n}{A}\cdot\underset{n\times 1}{\boldsymbol{x}(t)} + \underset{n\times m_u}{B}\cdot\underset{m_u\times 1}{\boldsymbol{u}(t)} + \underset{n\times m_f}{B_f}\cdot\underset{m_f\times 1}{\boldsymbol{f}(t)} \\ \underset{p\times 1}{\boldsymbol{y}(t)} = \underset{p\times n}{C}\cdot\underset{n\times 1}{\boldsymbol{x}(t)} + \underset{p\times m_u}{D}\cdot\underset{m_u\times 1}{\boldsymbol{u}(t)} + \underset{p\times m_f}{D_f}\cdot\underset{m_f\times 1}{\boldsymbol{f}(t)} \end{cases}$$



Residuals for fault diagnosis can be produced by **observers**, via a **state error** $\boldsymbol{e_x}(t)$ or an **output error** $\boldsymbol{e_y}(t)$

$$\boldsymbol{e_x}(t) \coloneqq \boldsymbol{x}(t) - \hat{\boldsymbol{x}}(t) \qquad \boldsymbol{e_y}(t) \coloneqq \boldsymbol{y}(t) - \hat{\boldsymbol{y}}(t)$$

Usually, the **output error is used**, since not all states in $\boldsymbol{x}(t)$ could be measured and the effects of output faults are more directly assessed

The observer-based framework to fault diagnosis

The idea of observer-based approaches is to **reconstruct** the **states** or the **outputs** of the system, using

- **Luenberger-like observers** in deterministic settings *(weighted output error as residual)*
- **Kalman-like filter** in stochastic settings *(weighted innovation error as residual)*

The **necessary** and **sufficient** condition for estimating the states with observers is that (C, A) **is observable**

However, we are **not** primarily **interested** in **estimating** the **states** of the system, but **only** its **outputs**. For this reason, **output observers** can be used

In this case, the **observer-based residual generator always exists**, since any input-output transfer function matrix has the *observable realization*

Outline

1. The observer-based framework
2. **Deterministic setting: functional observers**
3. Functional (output) observers for residual generation
4. Stochastic setting: Kalman-filter

Functional observers

Output observers are a specific type of **functional observers**, whose aim is to estimate a **linear function** $L \cdot \boldsymbol{x}(t)$ of the system state $\boldsymbol{x}(t)$. The functional observer form is:

$$\begin{cases} \underset{n_m \times 1}{\boldsymbol{m}(t+1)} = \underset{n_m \times n_m}{F} \cdot \underset{n_m \times 1}{\boldsymbol{m}(t)} + \underset{n_m \times p}{K} \cdot \underset{p \times 1}{\boldsymbol{y}(t)} + \underset{n_m \times m_u}{J} \cdot \underset{m_u \times 1}{\boldsymbol{u}(t)} \\ \underset{n_h \times 1}{\boldsymbol{h}(t)} = \underset{n_h \times n_m}{G} \cdot \underset{n_m \times 1}{\boldsymbol{m}(t)} + \underset{n_h \times p}{R} \cdot \underset{p \times 1}{\boldsymbol{y}(t)} + \underset{n_h \times m_u}{S} \cdot \underset{m_u \times 1}{\boldsymbol{u}(t)} \end{cases}$$

- $\boldsymbol{m} \in \mathbb{R}^{n_m \times 1}$ is the **states** vector
- $\boldsymbol{h} \in \mathbb{R}^{n_h \times 1}$ is the **outputs** vector

We want the **observer output** $\boldsymbol{h}(t)$ to provide an estimate of $L \cdot \boldsymbol{x}(t)$, so that

$$\lim_{t \to +\infty} [\underset{n_h \times 1}{\boldsymbol{h}(t)} - \underset{n_h \times n}{L} \cdot \underset{n \times 1}{\boldsymbol{x}(t)}] = \underset{n_h \times 1}{\boldsymbol{0}}$$

Functional observers

$$\lim_{t \to +\infty} [\underset{n_h \times 1}{\boldsymbol{h}(t)} - \underset{n_h \times n}{L} \cdot \underset{n \times 1}{\boldsymbol{x}(t)}] = \underset{n_h \times 1}{\boldsymbol{0}}$$

The matrix L defines ***any linear transformation of interest***, such as

- a state-feedback **control law** if $L = H_{\text{fbk}}$, with H_{fbk} a gain matrix so that

$$\boldsymbol{h}(t) \coloneqq \boldsymbol{u}(t) = H_{\text{fbk}} \boldsymbol{x}(t)$$

- the **system output** (without $D\boldsymbol{u}(t)$) i.e. $\boldsymbol{y}(t) - D\boldsymbol{u}(t)$ if $L = C$ so that

$$\boldsymbol{h}(t) \coloneqq \boldsymbol{y}(t) - D\boldsymbol{u}(t) = C\boldsymbol{x}(t)$$

Functional observers

To obtain $\boldsymbol{h}(t) \to L \cdot \boldsymbol{x}(t)$, we introduce an **auxiliary transformation matrix** T, so that

$$\lim_{t \to +\infty} [\underset{n_m \times 1}{\boldsymbol{m}(t)} - \underset{n_m \times n}{T} \cdot \underset{n \times 1}{\boldsymbol{x}(t)}] = \underset{n_m \times 1}{\boldsymbol{0}}$$

So that the **observer state** $\boldsymbol{m}(t)$ provides an estimate of a linear transformation T of the system state $\boldsymbol{x}(t)$. The matrix T can be chosen to:

- define a **reduced order observer**, when $n_m < n$
- obtain further properties of the estimate, such as the **decoupling of unknown inputs**

  → this is especially important in the context of fault diagnosis, where unknown inputs could be disturbances

Functional observers

We define the **functional state estimation error** $\boldsymbol{\varepsilon_m}(t)$ as

$$\underset{n_m \times 1}{\boldsymbol{\varepsilon_m}(t)} \coloneqq \underset{n_m \times 1}{\boldsymbol{m}(t)} - \underset{n_m \times n}{T} \cdot \underset{n \times 1}{\boldsymbol{x}(t)}$$

The recursive expression of $\boldsymbol{\varepsilon_m}(t)$ is

$$\underset{n_m \times 1}{\boldsymbol{\varepsilon_m}(t+1)} = \overset{n_m \times 1}{F} \cdot \underset{n_m \times n_m}{\boldsymbol{\varepsilon_m}(t)} + (\overset{n_m \times n}{FT} + \underset{n_m \times p}{KC} - \underset{n_m \times n}{TA}) \cdot \underset{n \times 1}{\boldsymbol{x}(t)} + (\underset{n_m \times m_u}{J} - \underset{n_m \times p}{TB} + \underset{n \times m_u}{KD}) \cdot \underset{m_u \times 1}{\boldsymbol{u}(t)}$$

It follows that

$$\boldsymbol{\varepsilon_m}(t+1) = F\boldsymbol{\varepsilon_m}(t) \quad \textbf{iff} \quad \begin{cases} TA = FT + KC \\ J = TB - KD \end{cases}$$

Then, the estimate $\boldsymbol{m}(t)$ approaches $T\boldsymbol{x}(t)$ in an *asymptotic* sense

$$\lim_{t \to +\infty} [\underset{n_m \times 1}{\boldsymbol{m}(t)} - \underset{n_m \times n}{T} \cdot \underset{n \times 1}{\boldsymbol{x}(t)}] = \mathbf{0} \quad \textbf{iff} \quad F \text{ has } \textbf{stable} \text{ eigenvalues}$$

Functional observers

Let's prove the recursion for $\boldsymbol{\varepsilon_m}(t)$. Start from its definition

$$\underset{n_m \times 1}{\boldsymbol{\varepsilon_m}(t)} \coloneqq \underset{n_m \times 1}{\boldsymbol{m}(t)} - \underset{n_m \times n}{T} \cdot \underset{n \times 1}{\boldsymbol{x}(t)}$$

Then:

$$\underset{n_m \times 1}{\boldsymbol{\varepsilon_m}(t+1)} = \overset{n_m \times 1}{\boldsymbol{m}(t+1)} - \underset{n_m \times n}{T}\overset{n \times 1}{\boldsymbol{x}(t+1)} = \underset{n_m \times n_m}{F}\overset{n_m \times 1}{\boldsymbol{z}(t)} + \underset{n_m \times p}{K}\overset{p \times 1}{\boldsymbol{y}(t)} + \underset{n_m \times m_u}{J}\overset{m_u \times 1}{\boldsymbol{u}(t)} - \underset{n_m \times n}{TA}\overset{n \times 1}{\boldsymbol{x}(t)} - \underset{n_m \times n}{TB}\overset{n \times m_u}{\boldsymbol{u}(t)}$$

$$= F\boldsymbol{m}(t) - FT\boldsymbol{x}(t) + FT\boldsymbol{x}(t) + K\boldsymbol{y}(t) + J\boldsymbol{u}(t) - TA\boldsymbol{x}(t) - TB\boldsymbol{u}(t)$$

$$= F\boldsymbol{\varepsilon_m}(t) + FT\boldsymbol{x}(t) + K\boldsymbol{y}(t) + J\boldsymbol{u}(t) - TA\boldsymbol{x}(t) - TB\boldsymbol{u}(t)$$

$$= F\boldsymbol{\varepsilon_m}(t) + FT\boldsymbol{x}(t) + K[C\boldsymbol{x}(t) + D\boldsymbol{u}(t)] + J\boldsymbol{u}(t) - TA\boldsymbol{x}(t) - TB\boldsymbol{u}(t)$$

$$= F\boldsymbol{\varepsilon_m}(t) + (FT + KC - TA)\boldsymbol{x}(t) + (J - TB + KD)\boldsymbol{u}(t)$$

Advanced material

Functional observers

Consider now the estimate provided by the **observer output** $\boldsymbol{h}(t)$. We define the **functional output estimation error** $\boldsymbol{\varepsilon_y}(t)$ as

$$\underset{n_h \times 1}{\boldsymbol{\varepsilon_y}(t)} \coloneqq \underset{n_h \times 1}{\boldsymbol{h}(t)} - \underset{n_h \times n}{L} \cdot \underset{n \times 1}{\boldsymbol{x}(t)}$$

Then

$$\boldsymbol{\varepsilon_y}(t) = \boldsymbol{h}(t) - L \cdot \boldsymbol{x}(t) = G\boldsymbol{m}(t) + S\boldsymbol{u}(t) + R\boldsymbol{y}(t) - L\boldsymbol{x}(t)$$

If previous functional state conditions hold, then $\boldsymbol{m}(t) \to T\boldsymbol{x}(t)$

$$= G[T\boldsymbol{x}(t)] + S\boldsymbol{u}(t) + R[C\boldsymbol{x}(t) + D\boldsymbol{u}(t)] - L\boldsymbol{x}(t)$$

$$= (GT + RC - L) \cdot \boldsymbol{x}(t) + (RD + S) \cdot \boldsymbol{u}(t)$$

Functional observers

$$\boldsymbol{\varepsilon_y}(t) = (GT + RC - L) \cdot \boldsymbol{x}(t) + (RD + S) \cdot \boldsymbol{u}(t)$$

We have that

$$\boldsymbol{\varepsilon_y}(t) = \mathbf{0} \quad \textbf{iif} \quad \begin{cases} L = GT + RC \\ S + RD = \mathbf{0} \end{cases}$$

Summarizing, the functional observer is able to produce an ***estimate of linear functions of the state*** if the following **Luenberger's conditions** hold

$$\begin{cases} F \text{ stable} \\ TA = FT + KC \\ J = TB - KD \end{cases} \qquad \begin{cases} L = GT + RC \\ S + RD = \mathbf{0} \end{cases}$$

Output observers: estimation of $y(t)$

If we want $\boldsymbol{h}(t)$ to provide an estimate of $\boldsymbol{y}(t)$, instead of $C\boldsymbol{x}(t) = \boldsymbol{y}(t) - D\boldsymbol{u}(t)$, after setting $L = C$, we have that

$$\begin{aligned}\boldsymbol{\varepsilon}_{\boldsymbol{y}}(t) \coloneqq \hat{\boldsymbol{y}}(t) - \boldsymbol{y}(t) &= \boldsymbol{h}(t) - \big(C\boldsymbol{x}(t) + D\boldsymbol{u}(t)\big) \\ &= G\boldsymbol{z}(t) + S\boldsymbol{u}(t) + R\boldsymbol{y}(t) - C\boldsymbol{x}(t) - D\boldsymbol{u}(t) \\ &= G[T\boldsymbol{x}(t)] + S\boldsymbol{u}(t) + R[C\boldsymbol{x}(t) + D\boldsymbol{u}(t)] - L\boldsymbol{x}(t) - D\boldsymbol{u}(t) \\ &= (GT + RC - L)\cdot\boldsymbol{x}(t) + (RD + S - D)\cdot\boldsymbol{u}(t)\end{aligned}$$

So that the Luenberger conditions on the outputs change to:

$$\boldsymbol{\varepsilon}_{\boldsymbol{y}}(t) = \boldsymbol{0} \quad \textbf{iif} \quad \begin{cases} L = GT + RC \\ \boxed{S + RD = D} \end{cases}$$

Luenberger observer as functional observer

Suppose that we are interested in full-order **state and outputs estimation.** Then:

- $\underset{n_m \times n}{T} = \underset{n}{I_n}$ and $\underset{n_h \times n}{L} = C$, so that $n_m = n$ and $n_h = p$
- $\boldsymbol{m}(t) \coloneqq \hat{\boldsymbol{x}}(t) \in \mathbb{R}^{n\times 1}$ is the **state** of the observer (that estimates the system state)
- $\boldsymbol{h}(t) \coloneqq C\hat{\boldsymbol{x}}(t) \in \mathbb{R}^{p\times 1}$ is the **output** of the observer (that estimates the system outputs minus $D\boldsymbol{u}(t)$)

If $\boldsymbol{m}(t) \coloneqq \hat{\boldsymbol{x}}(t)$, then the estimate of $C\boldsymbol{x}(t)$ can be obtained by $\boldsymbol{h}(t) = C\hat{\boldsymbol{x}}(t)$, so $R = S = 0$

Functional observer

$$\begin{cases} \underset{n_m\times 1}{\boldsymbol{m}(t+1)} = \underset{n_m\times n_m}{F}\cdot\underset{n_m\times 1}{\boldsymbol{m}(t)} + \underset{n_m\times p}{K}\cdot\underset{p\times 1}{\boldsymbol{y}(t)} + \underset{n_m\times m_u}{J}\cdot\underset{m_u\times 1}{\boldsymbol{u}(t)} \\ \underset{n_h\times 1}{\boldsymbol{h}(t)} = \underset{n_h\times n_m}{G}\cdot\underset{n_m\times 1}{\boldsymbol{m}(t)} + \underset{n_h\times p}{R}\cdot\underset{p\times 1}{\boldsymbol{y}(t)} + \underset{n_h\times m_u}{S}\cdot\underset{m_u\times 1}{\boldsymbol{u}(t)} \end{cases} \Rightarrow$$

Full-order state observer (Luenberger observer)

$$\begin{cases} \hat{\boldsymbol{x}}(t+1) = F\cdot\hat{\boldsymbol{x}}(t) + K\cdot\boldsymbol{y}(t) + J\cdot\boldsymbol{u}(t) \\ \boldsymbol{h}(t) = C\cdot\hat{\boldsymbol{x}}(t) \longrightarrow C = GT + RC = G \\ \hat{\boldsymbol{y}}(t) = \boldsymbol{h}(t) + D\cdot\boldsymbol{u}(t) \end{cases}$$

Luenberger observer as functional observer

Functional observer

$$\begin{cases} \underset{n_m\times 1}{\boldsymbol{m}(t+1)} = \underset{n_m\times n_m}{F}\cdot\underset{n_m\times 1}{\boldsymbol{m}(t)} + \underset{n_m\times p}{K}\cdot\underset{p\times 1}{\boldsymbol{y}(t)} + \underset{n_m\times m_u}{J}\cdot\underset{m_u\times 1}{\boldsymbol{u}(t)} \\ \underset{n_h\times 1}{\boldsymbol{h}(t)} = \underset{n_h\times n_m}{G}\cdot\underset{n_m\times 1}{\boldsymbol{m}(t)} + \underset{n_h\times p}{R}\cdot\underset{p\times 1}{\boldsymbol{y}(t)} + \underset{n_h\times m_u}{S}\cdot\underset{m_u\times 1}{\boldsymbol{u}(t)} \end{cases} \Rightarrow$$

Full-order state observer

$$\begin{cases} \hat{\boldsymbol{x}}(t+1) = F\cdot\hat{\boldsymbol{x}}(t) + K\cdot\boldsymbol{y}(t) + J\cdot\boldsymbol{u}(t) \\ \boldsymbol{h}(t) = C\cdot\hat{\boldsymbol{x}}(t) \\ \hat{\boldsymbol{y}}(t) = \boldsymbol{h}(t) + D\cdot\boldsymbol{u}(t) \end{cases}$$

Since $T = I_n$ the observer state error estimate $\boldsymbol{\varepsilon}_{\boldsymbol{m}}(t) = \boldsymbol{m}(t) - T\boldsymbol{x}(t)$ obeys the relations:

$$\lim_{t\to+\infty}\underset{n\times 1}{[\boldsymbol{m}(t)} - \underset{n\times n}{I_n}\cdot\underset{n\times 1}{\boldsymbol{x}(t)}] = \boldsymbol{0} \quad \textbf{iif} \quad \begin{cases} I_nA = FI_n + KC \\ J = I_nB - KD \end{cases} \Rightarrow \begin{cases} F = A - KC \\ J = B - KD \end{cases}$$

$\boldsymbol{m}(t) \coloneqq \hat{\boldsymbol{x}}(t)$

Luenberger observer as functional observer

The functional observer for full-order state estimation is the **Luenberger observer!**

$$\begin{cases} \hat{\boldsymbol{x}}(t+1) = (A - KC)\cdot\hat{\boldsymbol{x}}(t) + K\cdot\boldsymbol{y}(t) + (B - KD)\cdot\boldsymbol{u}(t) \\ \boldsymbol{h}(t) = C\cdot\hat{\boldsymbol{x}}(t) \\ \hat{\boldsymbol{y}}(t) = \boldsymbol{h}(t) + D\cdot\boldsymbol{u}(t) \end{cases}$$

The matrix K can be chosen by **pole placement** if (C, A) is observable

Outline

1. The observer-based framework
2. Deterministic setting: functional observers
3. **Functional (output) observers for residual generation**
4. Stochastic setting: Kalman-filter

Output observers for residuals generation

To generate **residuals**, we need to estimate the **system outputs**. To this end, let

$$\underset{n_h \times n}{L} = \underset{p \times n}{C} \quad \Rightarrow \quad \lim_{t \to +\infty} [\underset{p \times 1}{\boldsymbol{h}(t)} - \underset{p \times n}{L} \cdot \underset{n \times 1}{\boldsymbol{x}(t)}] = \lim_{t \to +\infty} [\boldsymbol{h}(t) - C \cdot \boldsymbol{x}(t)] = \underset{p \times 1}{\boldsymbol{0}}$$

The **outputs of the observer** $\boldsymbol{h}(t) \in \mathbb{R}^{n_h \times 1}$ produce an estimate of $\boldsymbol{y}(t) - D\boldsymbol{u}(t)$, where the dimension of the observer output is the dimension of the system outputs, i.e. $n_h = p$

The **estimate of system outputs with** $D\boldsymbol{u}(t)$ is: $\hat{\boldsymbol{y}}(t) = \boldsymbol{h}(t) + D\boldsymbol{u}(t)$

The **residual vector** $\boldsymbol{r}(t) \in \mathbb{R}^{q \times 1}$ is defined as:

$$\underset{q \times 1}{\boldsymbol{r}(t)} = \underset{q \times p}{W} \cdot [\underset{p \times 1}{\boldsymbol{y}(t)} - \underset{p \times 1}{\hat{\boldsymbol{y}}(t)}] \qquad W \neq \boldsymbol{0}$$

Output observers for residuals generation

The **residual** $\boldsymbol{r}(t)$ can be considered as the output of a **residual generator** observer:

$$\begin{aligned}
\underset{q \times 1}{\boldsymbol{r}(t)} = \underset{q \times p}{W} \cdot [\underset{p \times 1}{\boldsymbol{y}(t)} - \underset{p \times 1}{\hat{\boldsymbol{y}}(t)}] &= \underset{q \times p}{W} \cdot [\boldsymbol{y}(t) - \underset{p \times 1}{\boldsymbol{h}(t)} - \underset{p \times m_u}{D} \underset{m_u \times 1}{\boldsymbol{u}(t)}] \\
&= W \cdot [\boldsymbol{y}(t) - \underset{p \times n_m}{G}\boldsymbol{m}(t) - \underset{p \times m_u}{S}\boldsymbol{u}(t) - \underset{p \times p}{R}\boldsymbol{y}(t) - D\boldsymbol{u}(t)] \\
&= -WG \cdot \boldsymbol{m}(t) + [W - WR] \cdot \boldsymbol{y}(t) - W[S + D] \cdot \boldsymbol{u}(t) \\
&= \underset{q \times m}{L_1} \cdot \boldsymbol{m}(t) + \underset{q \times p}{L_2} \cdot \boldsymbol{y}(t) + \underset{q \times m_u}{L_3} \cdot \boldsymbol{u}(t)
\end{aligned}$$

where

$$\begin{cases} L_1 = -WG \\ L_2 = W(I - R) \\ L_3 = -W(S + D) \end{cases}$$

Output observers for residuals generation

Functional observer $\Rightarrow$ **Observer-based residual generator**

$$\begin{cases} \underset{n_m \times 1}{\boldsymbol{m}(t+1)} = \underset{n_m \times n_m}{F} \cdot \boldsymbol{m}(t) + \underset{n_m \times p}{K} \cdot \boldsymbol{y}(t) + \underset{n_m \times m_u}{J} \cdot \boldsymbol{u}(t) \\ \underset{p \times 1}{\boldsymbol{h}(t)} = G \cdot \boldsymbol{m}(t) + R \cdot \boldsymbol{y}(t) + S \cdot \boldsymbol{u}(t) \end{cases} \Rightarrow \begin{cases} \underset{n_m \times 1}{\boldsymbol{m}(t+1)} = \underset{n_m \times n_m}{F} \cdot \boldsymbol{m}(t) + \underset{n_m \times p}{K} \cdot \boldsymbol{y}(t) + \underset{n_m \times m_u}{J} \underset{m_u \times 1}{\boldsymbol{u}(t)} \\ \underset{q \times 1}{\boldsymbol{r}(t)} = \underset{q \times n_m}{L_1} \cdot \boldsymbol{m}(t) + \underset{q \times p}{L_2} \cdot \boldsymbol{y}(t) + \underset{q \times m_u}{L_3} \cdot \boldsymbol{u}(t) \end{cases}$$

The **Luenberger conditions** *are implied* by

$L_1 T + L_2 C = \boldsymbol{0}$ $\Rightarrow -WGT + WC - WRC = \boldsymbol{0} \Rightarrow -W(GT - C + RC) = \boldsymbol{0} \Rightarrow$ $L = C = GT + RC$

$L_3 + L_2 D = \boldsymbol{0}$ $\Rightarrow -WS - \cancel{WD} + \cancel{WD} - WRD = \boldsymbol{0} \Rightarrow -W(S + RD) = \boldsymbol{0} \Rightarrow$ $S + RD = \boldsymbol{0}$

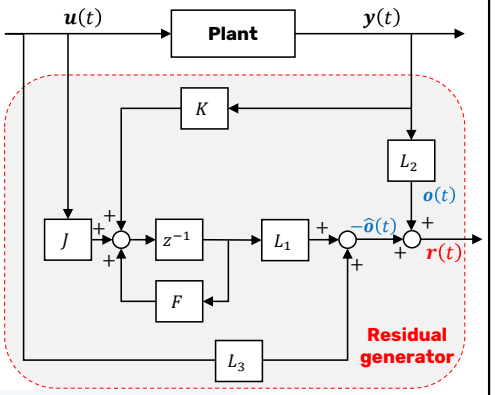
Output observers for residuals generation

Observer-based residual generator

$$\begin{cases} \boldsymbol{m}(t+1) = F \cdot \boldsymbol{m}(t) + K \cdot \boldsymbol{y}(t) + J \cdot \boldsymbol{u}(t) \\ \boldsymbol{r}(t) = L_1 \cdot \boldsymbol{m}(t) + L_2 \cdot \boldsymbol{y}(t) + L_3 \cdot \boldsymbol{u}(t) \\ \qquad = L_2 \cdot \boldsymbol{y}(t) - \hat{\boldsymbol{o}}(t) \end{cases}$$

$\hat{\boldsymbol{o}}(t) \coloneqq -L_1\boldsymbol{m}(t) - L_3\boldsymbol{u}(t)$ can be seen as an estimate of $\boldsymbol{o} \coloneqq L_2\boldsymbol{y}(t)$

$$\begin{cases} F \text{ stable} \\ TA = FT + KC \quad \text{I°} \\ J = TB - KD \quad \text{II°} \\ L_1T + L_2C = \mathbf{0} \quad \text{III°} \\ L_3 + L_2D = \mathbf{0} \quad \text{IV°} \end{cases}$$



Output observers for residuals generation

Given the observer-based residual generator

$$\begin{cases} \underset{n_m \times 1}{\boldsymbol{m}(t+1)} = \underset{n_m \times n_m}{F} \cdot \underset{n_m \times 1}{\boldsymbol{m}(t)} + \underset{n_m \times p}{K} \cdot \underset{p \times 1}{\boldsymbol{y}(t)} + \underset{n_m \times m_u}{J} \cdot \underset{m_u \times 1}{\boldsymbol{u}(t)} \\ \underset{q \times 1}{\boldsymbol{r}(t)} = \underset{q \times n_m}{L_1} \cdot \underset{n_m \times 1}{\boldsymbol{m}(t)} + \underset{q \times p}{L_2} \cdot \underset{p \times 1}{\boldsymbol{y}(t)} + \underset{q \times m_u}{L_3} \cdot \underset{m_u \times 1}{\boldsymbol{u}(t)} \end{cases}$$

its **computational form** can be expressed by the **transfer function matrix** $\boldsymbol{Q}(z)$ as

$$\underset{q \times 1}{\boldsymbol{r}(t)} = \underset{q \times (p+m_u)}{\boldsymbol{Q}(z)} \underset{(m_u+p) \times 1}{\begin{bmatrix} \boldsymbol{u}(t) \\ \boldsymbol{y}(t) \end{bmatrix}} = \underset{q \times m_u}{[\boldsymbol{Q}_u(z)} \quad \underset{q \times p}{\boldsymbol{Q}_y(z)}] \begin{bmatrix} \boldsymbol{u}(t) \\ \boldsymbol{y}(t) \end{bmatrix}$$

$$\boldsymbol{r}(t) = [L_1(zI - F)^{-1}J + L_3] \cdot \boldsymbol{u}(t) + [L_1(zI - F)^{-1}K + L_2] \cdot \boldsymbol{y}(t)$$

Residual generator (computational form)

Effects of faults

Let's now apply the observer-based **residual generator**

$$\begin{cases} \underset{n_m \times 1}{\boldsymbol{m}(t+1)} = \underset{n_m \times n_m}{F} \cdot \underset{n_m \times 1}{\boldsymbol{m}(t)} + \underset{n_m \times p}{K} \cdot \underset{p \times 1}{\boldsymbol{y}(t)} + \underset{n_m \times m_u}{J} \cdot \underset{m_u \times 1}{\boldsymbol{u}(t)} \\ \underset{q \times 1}{\boldsymbol{r}(t)} = \underset{q \times n_m}{L_1} \cdot \underset{n_m \times 1}{\boldsymbol{m}(t)} + \underset{q \times p}{L_2} \cdot \underset{p \times 1}{\boldsymbol{y}(t)} + \underset{q \times m_u}{L_3} \cdot \underset{m_u \times 1}{\boldsymbol{u}(t)} \end{cases}$$

to a general LTI **system** with **additive faults**

$$\begin{cases} \underset{n \times 1}{\boldsymbol{x}(t+1)} = \underset{n \times n}{A} \cdot \underset{n \times 1}{\boldsymbol{x}(t)} + \underset{n \times m_u}{B} \cdot \underset{m_u \times 1}{\boldsymbol{u}(t)} + \underset{n \times m_f}{B_f} \cdot \underset{m_f \times 1}{\boldsymbol{f}(t)} \\ \underset{p \times 1}{\boldsymbol{y}(t)} = \underset{p \times n}{C} \cdot \underset{n \times 1}{\boldsymbol{x}(t)} + \underset{p \times m_u}{D} \cdot \underset{m_u \times 1}{\boldsymbol{u}(t)} + \underset{p \times m_f}{D_f} \cdot \underset{m_f \times 1}{\boldsymbol{f}(t)} \end{cases}$$

Effects of faults

$$\begin{cases} F \text{ stable} \\ TA = FT + KC \quad \text{I°} \\ J = TB - KD \quad \text{II°} \\ L_1T + L_2C = \mathbf{0} \quad \text{III°} \\ L_3 + L_2D = \mathbf{0} \quad \text{IV°} \end{cases}$$

Consider again the functional state estimation error $\boldsymbol{\varepsilon_m}(t)$

$$\boldsymbol{\varepsilon_m}(t) \coloneqq \boldsymbol{m}(t) - T \cdot \boldsymbol{x}(t)$$

In presence of additive faults, we have that

$$\boldsymbol{\varepsilon_m}(t+1) = F\boldsymbol{\varepsilon_m}(t) + KD_f\boldsymbol{f}(t) - TB_f\boldsymbol{f}(t)$$

The **residual** reads as

$$\boldsymbol{r}(t) = L_1\boldsymbol{\varepsilon_m}(t) + L_2D_f\boldsymbol{f}(t)$$

Effects of faults

Let's prove the results of the previous slide.

$$\boldsymbol{\varepsilon_m}(t) \coloneqq \boldsymbol{m}(t) - T \cdot \boldsymbol{x}(t)$$

$$\begin{cases} F \text{ stable} \\ TA = FT + KC & \text{I°} \\ J = TB - KD & \text{II°} \\ L_1T + L_2C = \mathbf{0} & \text{III°} \\ L_3 + L_2D = \mathbf{0} & \text{IV°} \end{cases}$$

Then

$$\boldsymbol{\varepsilon_m}(t+1) = \boldsymbol{m}(t+1) - T\boldsymbol{x}(t+1) = F\boldsymbol{m}(t) + K\boldsymbol{y}(t) + J\boldsymbol{u}(t) - TA\boldsymbol{x}(t) - TB\boldsymbol{u}(t) - TB_f\boldsymbol{f}(t)$$

(I°)

$$= F\boldsymbol{m}(t) + K\left[C\boldsymbol{x}(t) + D\boldsymbol{u}(t) + D_f\boldsymbol{f}(t)\right] + J\boldsymbol{u}(t) - TA\boldsymbol{x}(t) - TB\boldsymbol{u}(t) - TB_f\boldsymbol{f}(t)$$

$$= -FT\boldsymbol{x}(t) + KCx(t) + KD\boldsymbol{u}(t) + KD_f\boldsymbol{f}(t) + J\boldsymbol{u}(t) - TB\boldsymbol{u}(t) - TB_f\boldsymbol{f}(t)$$

(II°)

$$= F[\boldsymbol{m}(t) - T\boldsymbol{x}(t)] + KD_f\boldsymbol{f}(t) - TB_f\boldsymbol{f}(t)$$

$$= F\boldsymbol{\varepsilon_m}(t) + KD_f\boldsymbol{f}(t) - TB_f\boldsymbol{f}(t)$$

Effects of faults

Consider now the residual $\boldsymbol{r}(t)$

$$\begin{cases} F \text{ stable} \\ TA = FT + KC & \text{I°} \\ J = TB - KD & \text{II°} \\ L_1T + L_2C = \mathbf{0} & \text{III°} \\ L_3 + L_2D = \mathbf{0} & \text{IV°} \end{cases}$$

$$\boldsymbol{r}(t) = L_1 \cdot \boldsymbol{m}(t) + L_2 \cdot \boldsymbol{y}(t) + L_3 \cdot \boldsymbol{u}(t)$$

$$= L_1\boldsymbol{m}(t) + L_2C\boldsymbol{x}(t) + L_2D\boldsymbol{u}(t) + L_2D_f\boldsymbol{f}(t) + L_3\boldsymbol{u}(t)$$

$$= L_1\left(T\boldsymbol{x}(t) + \boldsymbol{\varepsilon_z}(t)\right) + L_2C\boldsymbol{x}(t) + L_2D\boldsymbol{u}(t) + L_2D_f\boldsymbol{f}(t) + L_3\boldsymbol{u}(t)$$

$$= L_1T\boldsymbol{x}(t) + L_1\boldsymbol{\varepsilon_z}(t) + L_2C\boldsymbol{x}(t) + L_2D\boldsymbol{u}(t) + L_2D_f\boldsymbol{f}(t) + L_3\boldsymbol{u}(t)$$

$$= (L_1T + L_2C) \cdot \boldsymbol{x}(t) + (L_2D + L_3) \cdot \boldsymbol{u}(t) + L_1\boldsymbol{\varepsilon_z}(t) + L_2D_f\boldsymbol{f}(t)$$

(III°) (IV°)

$$= L_1\boldsymbol{\varepsilon_m}(t) + L_2D_f\boldsymbol{f}(t)$$

Effects of faults

The residual $\boldsymbol{r}(t)$ is thus governed by the following internal dynamics

$$\begin{cases} \underset{n_m \times 1}{\boldsymbol{\varepsilon_m}(t+1)} = \underset{n_m \times n_m}{F} \cdot \underset{n_m \times 1}{\boldsymbol{\varepsilon_m}(t)} + \underset{n_m \times p}{K} \underset{p \times m_f}{D_f} \cdot \underset{m_f \times 1}{\boldsymbol{f}(t)} - \underset{n_m \times n}{TB_f} \underset{n \times m_f}{\cdot \boldsymbol{f}(t)} \\ \underset{q \times 1}{\boldsymbol{r}(t)} = \underset{q \times n_m}{L_1} \cdot \underset{n_m \times 1}{\boldsymbol{\varepsilon_m}(t)} + \underset{q \times p}{L_2} \underset{p \times m_f}{D_f} \cdot \underset{m_f \times 1}{\boldsymbol{f}(t)} \end{cases}$$

The **internal form** of the residual generator can be expressed as

$$\boldsymbol{r}(t) = \left[L_1(zI - F)^{-1}\left(-TB_f + KD_f\right) + L_2D_f\right] \cdot \boldsymbol{f}(t)$$

Residual generator (internal form)

Luenberger observer as residual generator

For the Luenberger observer (see slides 15-17), it holds that

- $T = I_n$ and $L = C$, so that $n_m = n$ and $n_h = p$
- $\boldsymbol{m}(t) \coloneqq \hat{\boldsymbol{x}}(t) \in \mathbb{R}^{n \times 1}$ is the **state** of the observer (that estimates the system state)
- $\boldsymbol{h}(t) \coloneqq C\hat{\boldsymbol{x}}(t) \in \mathbb{R}^{p \times 1}$ is the **output** of the observer (that estimates the system outputs minus $D\boldsymbol{u}(t)$)
- $R = S = 0$
- $F = A - KC$
- $C = G$
- $J = B - KD$

Considering the matrices that define the output equation of an observer residual generator, it follows that

$$\begin{cases} L_1 = -WG = -WC \\ L_2 = W - WR = W \\ L_3 = -W(S + D) = -WD \end{cases}$$

Luenberger observer as residual generator

The **residual generator** based on the **Luenberger observer** is given by

$$\begin{cases} \underset{n\times 1}{\hat{\boldsymbol{x}}(t+1)} = \underset{n\times n}{(A-KC)}\cdot\underset{n\times 1}{\hat{\boldsymbol{x}}(t)} + \underset{n\times p}{K}\cdot\underset{p\times 1}{\boldsymbol{y}(t)} + \underset{n\times m_u}{(B-KD)}\cdot\underset{m_u\times 1}{\boldsymbol{u}(t)} \\ \underset{q\times 1}{\boldsymbol{r}(t)} = -\underset{q\times p}{W}\cdot\underset{p\times n}{C}\cdot\underset{n\times 1}{\hat{\boldsymbol{x}}(t)} + \underset{q\times p}{W}\cdot\underset{p\times 1}{\boldsymbol{y}(t)} - \underset{q\times p}{W}\underset{p\times m_u}{D}\cdot\underset{m_u\times 1}{\boldsymbol{u}(t)} = W\cdot[\boldsymbol{y}(t)-\hat{\boldsymbol{y}}(t)] \end{cases}$$

Hence, the **transfer function matrices** are given by

$$\begin{cases} \underset{q\times p}{\boldsymbol{Q}_y(z)} = \underset{q\times p}{W}\cdot\{-\underset{p\times n}{C}[zI_n-\underset{n\times n}{(A-KC)}]^{-1}\underset{n\times p}{K}+\underset{p\times p}{I_p}\} \\ \underset{q\times m_u}{\boldsymbol{Q}_u(z)} = \underset{q\times p}{W}\cdot\{-\underset{p\times n}{C}[zI_n-\underset{n\times n}{(A-KC)}]^{-1}\underset{n\times m_u}{(B-KD)}-\underset{p\times m_u}{D}\} \end{cases}$$

Luenberger observer as residual generator

The **residual** can be expressed as

$$\underset{q\times 1}{\boldsymbol{r}(t)} = \underset{q\times m_u}{\boldsymbol{Q}_u(z)}\underset{m_u\times 1}{\boldsymbol{u}(t)} + \underset{q\times p}{\boldsymbol{Q}_y(z)}\underset{p\times 1}{\boldsymbol{y}(t)}$$

Luenberger-observer residual generator (computational form)

$$= W\left\{\left[-C\big(zI_n-(A-KC)\big)^{-1}(B-KD)-D\right]\boldsymbol{u}(t)+\left[-C\big(zI_n-(A-KC)\big)^{-1}K+I_p\right]\boldsymbol{y}(t)\right\}$$

$$\boldsymbol{r}(t) = \left[-WC\big(zI-(A-KC)\big)^{-1}\big(-B_f+KD_f\big)+WD_f\right]\cdot\boldsymbol{f}(t)$$

$$= W\left[-C\big(zI-(A-KC)\big)^{-1}\big(-B_f+KD_f\big)+D_f\right]\cdot\boldsymbol{f}(t)$$

Luenberger-observer residual generator (internal form)

Luenberger observer as residual generator

Let's prove that what we found is a valid residual generator. Recall that it must hold that

$$\boldsymbol{Q}_u(z) = -\boldsymbol{Q}_y(z)\boldsymbol{G}(z)$$

Then, substituting the quantities just found:

$$W\left[-C\big(zI_n-(A-KC)\big)^{-1}(B-KD)-D\right] = -W\left[-C\big(zI_n-(A-KC)\big)^{-1}K+I_p\right][C(zI_n-A)^{-1}B+D]$$

Let $M \coloneqq \big(zI_n-(A-KC)\big)^{-1}$ and $N \coloneqq (zI_n-A)^{-1}$ be invertible. Then

$$W[-CM(B-KD)-D] = -W\big[-CMK+I_p\big]\cdot(CNB+D)$$

If this equation must hold, it is implied that

$$[CM(B-KD)+D] = \big[-CMK+I_p\big]\cdot(CNB+D)$$

Advanced material

Luenberger observer as residual generator

$$[CM(B-KD)+D] = \big[-CMK+I_p\big]\cdot(CNB+D)$$

$$CMB-CMKD+D = -CMKCNB-CMKD+CNB+D$$

$$CMB = -CMKCNB+CNB$$

Which implies that

$$M = -MKCN+N$$

$$M+MKCN = N \quad \Rightarrow \quad MN^{-1}+MKC = I_n \quad \Rightarrow \quad N^{-1}+KC = M^{-1}$$

Using the definitions $M \coloneqq \big(zI_n-(A-KC)\big)^{-1}$ and $N \coloneqq (zI_n-A)^{-1}$

$$zI_n-A+KC = zI_n-(A-KC)$$

proves the relation

Advanced material

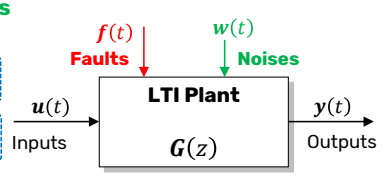
Outline

1. The observer-based framework
2. Deterministic setting: functional observers
3. Functional (output) observers for residual generation
4. **Stochastic setting: Kalman-filter**

Stochastic setting: Kalman-filter fault detection

Consider a LTI dynamical system with **faults** and **noises**

$$\begin{cases} \underset{n\times 1}{\boldsymbol{x}(t+1)} = \underset{n\times n}{A}\cdot\underset{n\times 1}{\boldsymbol{x}(t)} + \underset{n\times m_u}{B}\cdot\underset{m_u\times 1}{\boldsymbol{u}(t)} + \underset{n\times m_f}{B_f}\cdot\underset{m_f\times 1}{\boldsymbol{f}(t)} + \underbrace{\underset{n\times m_w}{B_w}\cdot\underset{m_w\times 1}{\boldsymbol{w}(t)}}_{\boldsymbol{v}_1(t)} \\ \underset{p\times 1}{\boldsymbol{y}(t)} = \underset{p\times n}{C}\cdot\underset{n\times 1}{\boldsymbol{x}(t)} + \underset{p\times m_u}{D}\cdot\underset{m_u\times 1}{\boldsymbol{u}(t)} + \underset{p\times m_f}{D_f}\cdot\underset{m_f\times 1}{\boldsymbol{f}(t)} + \underbrace{\underset{p\times m_w}{D_w}\cdot\underset{m_w\times 1}{\boldsymbol{w}(t)}}_{\boldsymbol{v}_2(t)} \end{cases}$$



$$\boldsymbol{v}_1(t) \sim \mathrm{GWN}(\mathbf{0}, V_1 = B_w B_w^\top), \qquad \boldsymbol{v}_2(t) \sim \mathrm{GWN}(\mathbf{0}, V_2 = D_w D_w^\top)$$

$$\mathbb{E}\begin{bmatrix}\boldsymbol{v}_1(t)\\ \boldsymbol{v}_2(s)\end{bmatrix} = \begin{bmatrix}V_1 & V_{12}\\ V_{12}^\top & V_2\end{bmatrix}\delta_{ts} \qquad \boldsymbol{v}_1(t), \boldsymbol{v}_2(t) \perp \boldsymbol{x}(0) \qquad \mathbb{E}[\boldsymbol{x}(0)] = \boldsymbol{x}^0 \qquad \mathbb{E}[(\boldsymbol{x}(0)-\boldsymbol{x}^0)(\boldsymbol{x}(0)-\boldsymbol{x}^0)^\top] = P_0^0$$

Under these assumptions, the **Kalman filter** delivers an **innovations sequence** $\boldsymbol{\varepsilon}_y(t) \coloneqq \boldsymbol{y}(t) - \hat{\boldsymbol{y}}(t|t-1)$ which is a **white Gaussian noise** process (if the model is perfect)

Advanced material

Stochastic setting: Kalman-filter fault detection

Any **deviation** of the **innovation** sequence from its **nominal statistical properties** can be used for fault detection

Kalman predictor/filter:

$$\begin{cases} \hat{\boldsymbol{x}}(t|t-1) = A(t)\hat{\boldsymbol{x}}(t-1|t-1) + B\boldsymbol{u}(t) \\ \hat{\boldsymbol{x}}(t|t) = \hat{\boldsymbol{x}}(t|t-1) + K_0(t)\boldsymbol{\varepsilon}_y(t) \\ \hat{\boldsymbol{y}}(t|t-1) = C(t)\hat{\boldsymbol{x}}(t|t-1) + D\boldsymbol{u}(t) \end{cases}$$

$$\hat{\boldsymbol{x}}(1|0) = \boldsymbol{x}^0$$
$$P_0(1|0) = P_0^0$$

$$\begin{cases} \boldsymbol{\varepsilon}_y(t) = \boldsymbol{y}(t) - \hat{\boldsymbol{y}}(t|t-1) \\ K_0(t) = P_0(t|t-1)C^\top(t)\mathrm{E}(t)^{-1} \\ \mathrm{E}(t) = C(t)P_0(t|t-1)C^\top(t) + V_2 \\ P_0(t|t-1) = A(t)P_0(t-1|t-1)A(t)^\top + V_1 \\ P_0(t|t) = \big(I_n - K_0(t)C(t)\big)P_0(t|t-1) \end{cases}$$

$P_0(t|t)$ is the covariance matrix of the filtering estimate $\hat{\boldsymbol{x}}(t|t)$

Advanced material

Stochastic setting: Kalman-filter fault detection

Significant properties of the Kalman filter are (under Gaussianity assumptions and model correctness):

- The state prediction covariance $P_0(t|t-1) \coloneqq \mathbb{E}\left[\big(\boldsymbol{x}(t) - \hat{\boldsymbol{x}}(t|t-1)\big)\big(\boldsymbol{x}(t) - \hat{\boldsymbol{x}}(t|t-1)\big)^\top\right]$ **is minimum**
- The **innovation process** $\boldsymbol{\varepsilon}_y(t) \coloneqq \boldsymbol{y}(t) - C(t)\hat{\boldsymbol{x}}(t|t-1) - D\boldsymbol{u}(t)$ is a **white Gaussian** process with covariance

$$\mathbb{E}\big[\boldsymbol{\varepsilon}_y(t)\boldsymbol{\varepsilon}_y^\top(t)\big] \coloneqq \mathrm{E}(t) = C(t)P_0(t|t-1)C^\top(t) + V_2$$

The idea is to **use the innovation** process $\boldsymbol{\varepsilon}_y(t)$ as **residual** $\boldsymbol{r}(t)$, using its nominal statistical properties to detect faults and setting a threshold

Advanced material

Stochastic setting: Kalman-filter fault detection

The **Kalman filter residual generator** is usually built with the **steady-state predictor** *(not the filter)* Kalman gain $\bar{K} = A\bar{P}C^{\top}(C\bar{P}C^{\top} + V_2)^{-1}$ (assuming $V_{12} = \mathbf{0}$) leading to:

$$\begin{cases} \underset{n\times 1}{\hat{\boldsymbol{x}}(t+1|t)} = (A - \bar{K}C)\cdot \underset{n\times n}{\hat{\boldsymbol{x}}(t|t-1)} + \underset{n\times p}{\bar{K}}\cdot \underset{p\times 1}{\boldsymbol{y}(t)} + (B - \bar{K}D)\cdot \underset{n\times m_u}{\boldsymbol{u}(t)} \\ \underset{q\times 1}{\boldsymbol{r}(t)} = \underset{q\times p}{W}[\boldsymbol{y}(t) - \hat{\boldsymbol{y}}(t|t-1)] = W[\boldsymbol{y}(t) - (C\cdot \hat{\boldsymbol{x}}(t|t-1) + D\boldsymbol{u}(t))] \end{cases}$$

The **covariance matrix** of the **residuals** (in the nominal case) is

$$\mathbb{E}[\boldsymbol{r}(t)\boldsymbol{r}^{\top}(t)] \coloneqq \underset{q\times q}{\mathbf{R}} = W(C\bar{P}C^{\top} + V_2)W^{\top}$$

with $\bar{P}$ the steady-state **prediction** *(not filtering)* **covariance** matrix

Stochastic setting: Kalman-filter fault detection

The **residuals evaluation** is made by computing the test statistics $\xi(t)$

$$\xi(t) = \underset{1\times q}{\boldsymbol{r}^{\top}(t)}\underset{q\times q}{\mathbf{R}^{-1}(t)}\underset{q\times 1}{\boldsymbol{r}(t)}$$

The statistics $\xi(t)$, *after initial condition transient*, is distributed as a χ^2**-distribution** with q **degrees of freedom**

The threshold η is determined using the χ^2-distribution and specifying a significance level **(false alarms rate)** α so that

$$\text{Prob}[\xi(t) \leq \eta \;\mid\; \text{healthy system}] = 1 - \alpha$$

Kalman filter as residual generator

The steady-state **Kalman filter and predictor** can be written as

$$\begin{cases} \hat{\boldsymbol{x}}(t+1|t) = (A - \bar{K}C)\cdot \hat{\boldsymbol{x}}(t|t-1) + \bar{K}\cdot \boldsymbol{y}(t) + (B - \bar{K}D)\cdot \boldsymbol{u}(t) \\ \hat{\boldsymbol{x}}(t|t) = (I_n - \bar{K}_0C)\cdot \hat{\boldsymbol{x}}(t|t-1) + \bar{K}_0\cdot \boldsymbol{y}(t) - \bar{K}_0D\cdot \boldsymbol{u}(t) \\ \hat{\boldsymbol{y}}(t|t-1) = C\cdot \hat{\boldsymbol{x}}(t|t-1) + D\cdot \boldsymbol{u}(t) \end{cases}$$

- $\bar{K} = A\bar{P}C^{\top}(C\bar{P}C^{\top} + V_2)^{-1}$ is the steady-state **predictor** gain (when $V_{12} = \mathbf{0}$)
- $\bar{K}_0 = \bar{P}C^{\top}(C\bar{P}C^{\top} + V_2)^{-1}$ is the steady-state **filter** gain
- $\bar{P}$ is the steady state **prediction error** covariance matrix

Kalman filter as residual generator

If we define a new output measurements set as

$$\underset{(n+p)\times 1}{\boldsymbol{z}(t)} \coloneqq \begin{bmatrix} \underset{n\times 1}{\hat{\boldsymbol{x}}(t|t)} \\ \underset{p\times 1}{\hat{\boldsymbol{y}}(t|t-1)} \end{bmatrix}$$

then

$$\begin{cases} \underset{n\times n}{\hat{\boldsymbol{x}}(t+1|t)} = (A - \bar{K}C)\cdot \hat{\boldsymbol{x}}(t|t-1) + \underset{n\times p}{\bar{K}}\cdot \boldsymbol{y}(t) + (B - \bar{K}D)\cdot \boldsymbol{u}(t) \\ \underset{(n+p)\times 1}{\boldsymbol{z}(t)} = \underset{(n+p)\times n}{\begin{bmatrix} I_n - \bar{K}_0C \\ C \end{bmatrix}}\underset{n\times 1}{\hat{\boldsymbol{x}}(t|t-1)} + \underset{(n+p)\times p}{\begin{bmatrix} \bar{K}_0 \\ \mathbf{0} \end{bmatrix}}\cdot \underset{p\times 1}{\boldsymbol{y}(t)} + \underset{(n+p)\times m_u}{\begin{bmatrix} -\bar{K}_0D \\ D \end{bmatrix}}\cdot \underset{m_u\times 1}{\boldsymbol{u}(t)} \end{cases}$$

Example: Luenberger observer and Kalman filter

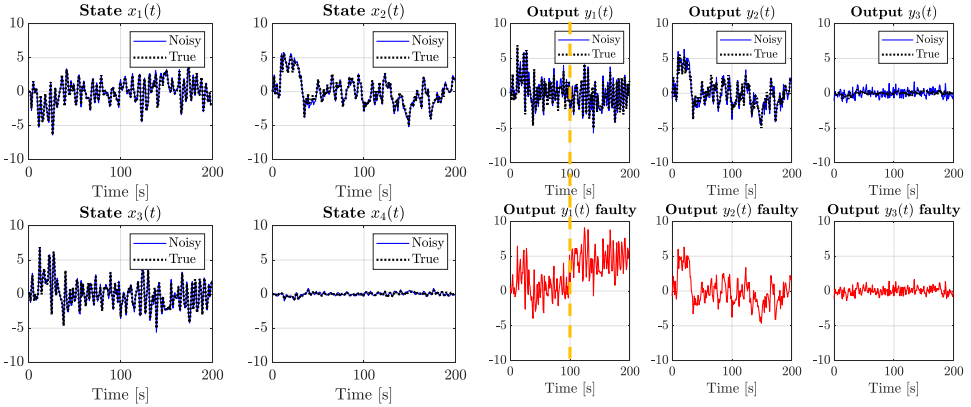
Consider the following stochastic MIMO system with $m_u = 2, p = 3, m_f = 1$, sampled at $T_s = 1$ s. Assume there are $m_w = 7$ noises acting on states and outputs

$$\underset{n\times n}{A} = \underset{4\times 4}{\begin{bmatrix} 0.5 & -0.7 & 0.7 & 0 \\ 0 & 0.8 & 0.06 & 0 \\ -1 & 0 & 0 & 0.1 \\ 0 & 0 & -0.1 & 0.4 \end{bmatrix}} \quad \underset{n\times m_u}{B} = \underset{4\times 2}{\begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}} \quad \underset{p\times n}{C} = \underset{3\times 4}{\begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}} \quad \underset{p\times m_u}{D} = \underset{3\times 2}{\begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}}$$

$$\underset{n\times m_f}{B_f} = \underset{2\times 1}{\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}} \quad \underset{n\times m_f}{D_f} = \underset{2\times 1}{\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}} \quad \underset{n\times m_w}{B_w} = \underset{4\times 7}{[0.1I_4 \;\; \mathbf{0}_{4\times 3}]} \quad \underset{p\times m_w}{D_w} = \underset{4\times 7}{[\mathbf{0}_{3\times 4} \;\; 0.5I_3]}$$

Collect $N = 200$ data. The inputs are white noises with zero mean and variance 1. A step-like fault of amplitude 5 acts on the system at time $t = 100$ s.

Example: Luenberger observer and Kalman filter

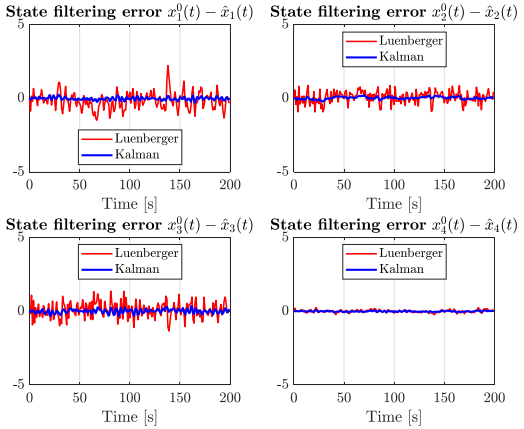


Example: Luenberger observer and Kalman filter

The plots depict the state filtering errors (in healthy case)

A steady-state Kalman **filter** was used to generate state estimates $\hat{x}(t|t)$

The Kalman filter does a better job in filtering the state

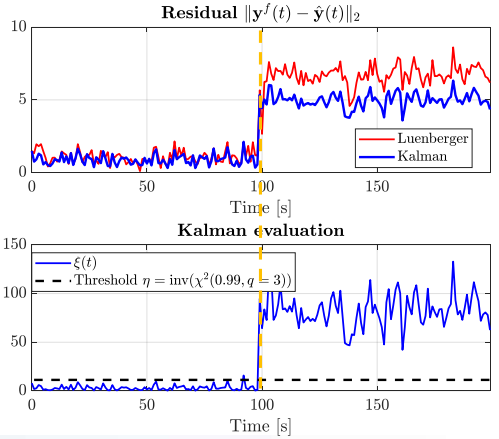


Example: Luenberger observer and Kalman filter

A steady-state Kalman **predictor** is used to generate residuals

Both observers perform well in terms of fault detection

The threshold η for the Kalman filter evaluation $\xi(t)$ has been set computing the 99-th percentile of a $\chi^2(q = 2)$ distribution



UNIVERSITÀ DEGLI STUDI DI BERGAMO | Dipartimento di Ingegneria Gestionale, dell'Informazione e della Produzione

UNIVERSITÀ DEGLI STUDI DI BERGAMO | Dipartimento di Ingegneria Gestionale, dell'Informazione e della Produzione

ADAPTIVE LEARNING, ESTIMATION AND SUPERVISION OF DYNAMICAL SYSTEMS (ALES)

Lecture 13: Parameters estimation fault detection

Master Degree in COMPUTER ENGINEERING

Data Science and Data Engineering Curriculum

SPEAKER
Prof. Mirko Mazzoleni

PLACE
University of Bergamo

Syllabus

1. **Recursive and adaptive identification**
   - 1.1 Recursive ARX estimation (RLS)
   - 1.2 Least Mean Squares (LMS)
   - 1.3 Instrumental Variables (IV)
2. **Closed-loop identification**
3. **Subspace and MIMO identification**
   - 3.1 Singular Value Decomposition
   - 3.2 Impulse data: Ho-Kalman, Kung algorithms
   - 3.3 Generic I/O data: the MOESP algorithm
4. **Supervision of dynamical systems**
   - 4.1 Introduction to fault diagnosis
   - 4.2 Model-based fault diagnosis
   - 4.3 Parity space approaches
   - 4.4 Observer-based approaches
   - 4.5 Signal-based fault diagnosis
   - 4.6 Knowledge-based fault diagnosis

CASE STUDIES

- Virtual Reference Feedback Tuning
- Nuclear particles classification
- Leak detection in an industrial valve
- Bearing fault identification

Outline

1. The parameters estimation framework
2. Recursive Least Squares estimation for discrete-time MIMO systems
3. Parameters estimation with Extended Kalman Filter
4. Parameters estimation for continuous-time systems

Outline

1. **The parameters estimation framework**
2. Recursive Least Squares estimation for discrete-time MIMO systems
3. Parameters estimation with Extended Kalman Filter
4. Parameters estimation for continuous-time systems

The parameters estimation framework

The **parameters estimation** approach to fault diagnosis is based on the assumption that faults are reflected in the **physical** system **parameters** such as *friction*, *mass*, *viscosity*, *resistance*, *inductance*, *capacitance*, etc

Assume that there is a discrepancy between the true system and its model due to a **change in the system parameters**

$$\boldsymbol{G}_u^0(z) = \boldsymbol{G}_u(z) + \boldsymbol{\Delta G}(z)$$

The parameters estimation approach is a simple strategy for tackling **multiplicative** (parameters) **faults**

The parameters estimation framework

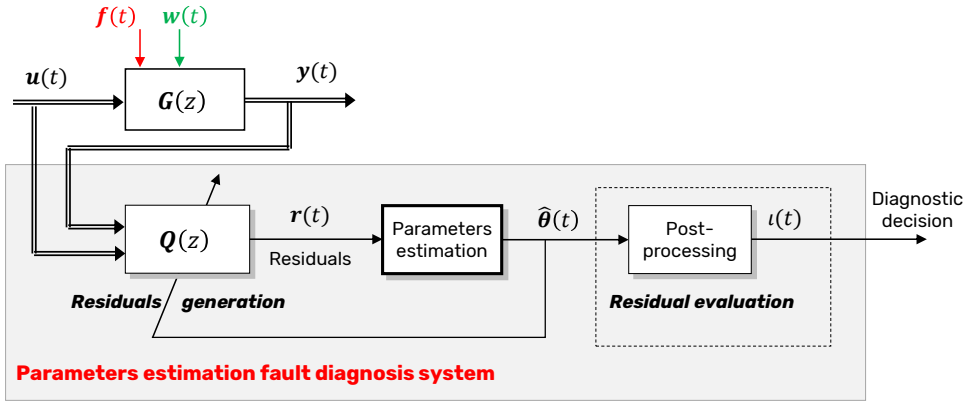
The idea is:

- Obtain a system model in **healthy conditions**, and the values of its parameters $\boldsymbol{\theta}(0)$
- Continuously **re-estimate on-line** the parameters $\widehat{\boldsymbol{\theta}}(t)$. Any substantial difference $\widehat{\boldsymbol{\theta}}(t) - \boldsymbol{\theta}(0)$ in the parameters values is a symptom of a fault

Residuals can be defined as

$$\boldsymbol{r}(t) = \widehat{\boldsymbol{\theta}}(t) - \boldsymbol{\theta}(0) \quad \textbf{or} \quad \boldsymbol{r}(t) = \boldsymbol{y}(t) - \hat{\boldsymbol{y}}\left(t; \widehat{\boldsymbol{\theta}}(t)\right)$$

The parameters estimation framework



The parameters estimation framework

In case of **SISO systems**, the most model assumption for parameters estimation is the use of an **ARX model structure**, which can be recursively estimated by **RLS** (recursive least squares) with **forgetting factor**

$$\mathcal{S}: y(t) = a_1 y(t-1) + \cdots + a_{n_a} y(t-n_a) + b_0 u(t-1) + \cdots + b_{n_b} u(t-n_b-1) + e(t)$$

$$= \underset{1\times d}{\boldsymbol{\varphi}^\top(t)} \underset{d\times 1}{\boldsymbol{\theta}} + e(t)$$

- $\boldsymbol{\varphi}(t) = [y(t-1) \quad \cdots \quad y(t-n_a) \quad u(t-1) \quad \cdots \quad u(t-n_b-1)]^\top$
- $\boldsymbol{\theta} = [a_1 \quad \cdots \quad a_{n_a} \quad b_0 \quad \cdots \quad b_{n_b}]^\top$
- $e(t) \sim \mathrm{WN}(0, \lambda^2)$ is an unknown white-noise signal

Alternatively, **recursive PEM** or **recursive IV** methods can be used to obtain correct estimates when the true system does not have an ARX structure

The parameters estimation framework

Remark

If input and output data do not have zero mean, a constant c can be added and estimated along with the model parameters

$$\mathcal{S}: y(t) = a_1 y(t-1) + \cdots + a_{n_a} y(t-n_a) + b_0 u(t-1) + \cdots + b_{n_b} u(t-n_b-1) + e(t) + c$$

$$= \boldsymbol{\varphi}^\top(t)\boldsymbol{\theta} + e(t)$$

- $\boldsymbol{\varphi}(t) = [y(t-1) \quad \cdots \quad y(t-n_a) \quad u(t-1) \quad \cdots \quad u(t-n_b-1) \quad 1]^\top$
- $\boldsymbol{\theta} = [a_1 \quad \cdots \quad a_{n_a} \quad b_0 \quad \cdots \quad b_{n_b} \quad c]^\top$
- $e(t) \sim \mathrm{WN}(0, \lambda^2)$ is an unknown white-noise signal

The parameters estimation framework

In the case of **MIMO** systems, there are three main alternatives:

1. Employ a **state-space** formulation to estimate parameters as states with a Kalman filter
2. Employ a **MIMO RLS** scheme. This allows also to modify the estimation with **IVs**
3. If a model is **(partly) known**, employ an **Extended Kalman filter (EKF)** to estimate parameters along with system states

Outline

The matrix polynomial I/O representation

How can we represent a **MIMO** $\mathrm{ARX}(n_a, n_b)$ model? Consider an $\mathrm{ARX}(n_a = 1, n_b = 1)$ model with $m_u = 3$ inputs and $p = 2$ outputs. The model requires **2 input/output equations**:

$$\begin{cases} y_1(t) = a_1^{11} y_1(t-1) + a_1^{12} y_2(t-1) + \\ \quad + b_0^{11} u_1(t-1) + b_0^{12} u_2(t-1) + b_0^{13} u_3(t-1) + \\ \quad + b_1^{11} u_1(t-2) + b_1^{12} u_2(t-2) + b_1^{13} u_3(t-2) + e_1(t) \\ y_2(t) = a_1^{21} y_1(t-1) + a_1^{22} y_2(t-1) + \\ \quad + b_0^{21} u_1(t-1) + b_0^{22} u_2(t-1) + b_0^{23} u_3(t-1) + \\ \quad + b_1^{21} u_1(t-2) + b_1^{22} u_2(t-2) + b_1^{23} u_3(t-2) + e_2(t) \end{cases}$$

$a^{\text{to output } i - \text{from output } j}_{\text{lag number } k}$

$b^{\text{to output } i - \text{from input } j}_{\text{lag number } k-1}$

The matrix polynomial I/O representation

The $\mathrm{ARX}(1,1)$ system can be expressed in a matrix form, as follows:

$$\underset{p\times 1}{\boldsymbol{y}(t)} = \underset{p\times p}{A_1}\underset{p\times 1}{\boldsymbol{y}(t-1)} + \underset{p\times m_u}{B_0}\underset{m_u\times 1}{\boldsymbol{u}(t-1)} + \underset{p\times m_u}{B_1}\underset{m_u\times 1}{\boldsymbol{u}(t-2)} + \underset{p\times 1}{\boldsymbol{e}(t)}$$

For instance:

$$\underset{p\times p}{A_1} = \underset{2\times 2}{\begin{bmatrix} a_1^{11} & a_1^{12} \\ a_1^{21} & a_1^{22} \end{bmatrix}} \qquad \underset{p\times m_u}{B_0} = \underset{2\times 3}{\begin{bmatrix} b_0^{11} & b_0^{12} & b_0^{13} \\ b_0^{21} & b_0^{22} & b_0^{23} \end{bmatrix}} \qquad \underset{p\times m_u}{B_1} = \underset{2\times 3}{\begin{bmatrix} b_1^{11} & b_1^{12} & b_1^{13} \\ b_1^{21} & b_1^{22} & b_1^{23} \end{bmatrix}}$$

$$\underset{p\times 1}{\boldsymbol{y}(t)} = \underset{2\times 1}{\begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix}} \qquad \underset{m_u\times 1}{\boldsymbol{u}(t)} = \underset{3\times 1}{\begin{bmatrix} u_1(t) \\ u_2(t) \\ u_3(t) \end{bmatrix}} \qquad \underset{p\times 1}{\boldsymbol{e}(t)} = \underset{2\times 1}{\begin{bmatrix} e_1(t) \\ e_2(t) \end{bmatrix}}$$

The matrix polynomial I/O representation

Using the **operatorial notation** one obtains the more compact description:

$$\underset{p\times p}{A(z)}\underset{p\times 1}{\boldsymbol{y}(t)} = \underset{p\times m_u}{B(z)}\underset{m_u\times 1}{\boldsymbol{u}(t-1)} + \underset{p\times 1}{\boldsymbol{e}(t)}$$

where $A(z) = I_p - A_1 z^{-1}$, $B(z) = B_0 + B_1 z^{-1}$ are **matrix polynomials**

This representation is analogous to the polynomial representation for SISO ARMAX models. Consider the SISO $\mathrm{ARX}(1,1)$ model $y(t) = a_1 y(t-1) + b_0 u(t-1) + b_1 u(t-2) + e(t)$. Then,

$$A(z)y(t) = B(z)u(t-1) + e(t)$$

where $A(z) = 1 - a_1 z^{-1}$, $B(z) = b_0 + b_1 z^{-1}$

Estimation of MIMO ARX models

Learning the $\mathrm{ARX}(n_a, n_b)$ model can still be tackled as a **linear regression problem:**

$$\underset{p\times 1}{\boldsymbol{y}(t)} = \underset{p\times (pd)}{\Psi(t)^\top}\underset{(pd)\times 1}{\boldsymbol{\theta}} + \underset{p\times 1}{\boldsymbol{e}(t)}$$

In our example:

$$\underset{p\times (pd)}{\Psi(t)^\top} = \underset{2\times 16}{\begin{bmatrix} \underset{1\times d}{\boldsymbol{\varphi}(t)^\top} & \cdots & \underset{1\times d}{\mathbf{0}} \\ \vdots & \ddots & \vdots \\ \underset{1\times d}{\mathbf{0}} & \cdots & \underset{1\times d}{\boldsymbol{\varphi}(t)^\top} \end{bmatrix}}$$

$$\underset{1\times (n_a\cdot p + n_b\cdot m_u)}{\boldsymbol{\varphi}(t)^\top} = \underset{1\times d}{[\boldsymbol{y}(t-1)^\top}\underset{1\times p}{} \quad \underset{1\times m_u}{\boldsymbol{u}(t-1)^\top} \quad \underset{1\times m_u}{\boldsymbol{u}(t-2)^\top}] \qquad 1\times(1\cdot 2 + 2\cdot 3) = 1\times 8$$

$$\underset{(pd)\times 1}{\boldsymbol{\theta}} = \underset{(2\cdot 8)\times 1 = 16\times 1}{\begin{bmatrix} \underset{8\times 1}{\boldsymbol{\theta}_1}^{d\times 1} \\ \underset{8\times 1}{\boldsymbol{\theta}_2}^{d\times 1} \end{bmatrix}}$$

$$\underset{p\times d}{\begin{bmatrix} (\boldsymbol{\theta}_1)^\top \\ (\boldsymbol{\theta}_2)^\top \end{bmatrix}} = \underset{p\times p}{[A_1}\ \underset{p\times m_u}{B_0}\ \underset{p\times m_u}{B_1}] = \underset{2\times 8}{\left[\begin{array}{cc|ccc|ccc} a_1^{11} & a_1^{12} & b_0^{11} & b_0^{12} & b_0^{13} & b_1^{11} & b_1^{12} & b_1^{13} \\ a_1^{21} & a_1^{22} & b_0^{21} & b_0^{22} & b_0^{23} & b_1^{21} & b_1^{22} & b_1^{23} \end{array}\right]}$$

A_1 B_0 B_1

Estimation of MIMO ARX models

In view of this reformulation, we are able to address the parameter estimation problem using **Least Squares** type algorithms as in the scalar case:

$$\underset{pd\times 1}{\widehat{\boldsymbol{\theta}}_N} = \left(\sum_{t=1}^{N} \underset{pd\times p}{\Psi(t)} \cdot \underset{p\times pd}{\Psi(t)^\top}\right)^{-1} \cdot \sum_{t=1}^{N} \underset{pd\times p}{\Psi(t)} \cdot \underset{p\times 1}{\boldsymbol{y}(t)}$$

$$\underset{pd\times 1}{\widehat{\boldsymbol{\theta}}_N} = \underset{pd\times pN}{(\boldsymbol{\Psi}^\top\boldsymbol{\Psi})^{-1}} \cdot \underset{pd\times pN}{\boldsymbol{\Psi}^\top} \cdot \underset{pN\times 1}{\boldsymbol{Y}} \qquad \boldsymbol{Y} = \begin{bmatrix} \boldsymbol{y}(1) \\ \vdots \\ \boldsymbol{y}(N) \end{bmatrix} \qquad \underset{pN\times pd}{\boldsymbol{\Psi}} = \begin{bmatrix} \underset{p\times pd}{\Psi(1)^\top} \\ \vdots \\ \underset{p\times pd}{\Psi(N)^\top} \end{bmatrix}$$

Estimation of MIMO ARX models

Notice that **the individual components** of $\boldsymbol{y}(t)$ can be described by the **independent parameter** vectors $\boldsymbol{\theta}_j \in \mathbb{R}^{d\times 1}$, $j = 1, \dots, p$:

$$\underset{1\times 1}{y_j(t)} = \underset{1\times d}{\boldsymbol{\varphi}(t)^\top}\underset{d\times 1}{\boldsymbol{\theta}_j} + \underset{1\times 1}{e_j(t)}$$

where $y_j(t)$ is the output of a **MISO** (multiple input single output) system, so that the proposed method is equivalent to applying p times the LS formula to a **scalar model:**

$$\underset{d\times 1}{\widehat{\boldsymbol{\theta}}_{j_N}} = \left(\sum_{t=1}^{N} \underset{d\times 1}{\boldsymbol{\varphi}(t)}\underset{1\times d}{\boldsymbol{\varphi}(t)^\top}\right)^{-1} \cdot \sum_{t=1}^{N} \underset{d\times 1}{\boldsymbol{\varphi}(t)}\underset{1\times 1}{y_j(t)} \qquad j = 1, \dots, p$$

Estimation of MIMO ARX models

That is

$$\underset{d\times 1}{\widehat{\boldsymbol{\theta}}_{j_N}} = \underset{d\times d}{(\Phi^\top\Phi)^{-1}}\underset{d\times N}{\Phi^\top}\underset{N\times 1}{Y_j} \qquad j = 1, \dots, p$$

$$\underset{N\times 1}{Y_j} = \begin{bmatrix} \underset{1\times 1}{y_j(1)^\top} \\ \vdots \\ \underset{1\times 1}{y_j(N)^\top} \end{bmatrix} \qquad \underset{N\times d}{\Phi} = \begin{bmatrix} \underset{1\times d}{\boldsymbol{\varphi}(1)^\top} \\ \vdots \\ \underset{1\times d}{\boldsymbol{\varphi}(N)^\top} \end{bmatrix}$$

Remark

The **RLS (or RIV) estimate** can be computed for each $\widehat{\boldsymbol{\theta}}_{j_N}$ as in the scalar case, or by adequately substituting $\Psi(t)$ and $\boldsymbol{y}(t)$ in place of $\boldsymbol{\varphi}(t)$ and $y(t)$ respectively (see Slide 27 Lecture 2)

Estimation of MIMO ARX models

A further alternative way is to define

$$\underset{N\times p}{Y} = \begin{bmatrix} \underset{1\times p}{\boldsymbol{y}(1)^\top} \\ \vdots \\ \underset{1\times p}{\boldsymbol{y}(N)^\top} \end{bmatrix} \qquad \underset{N\times d}{\Phi} = \begin{bmatrix} \underset{1\times d}{\boldsymbol{\varphi}(1)^\top} \\ \vdots \\ \underset{1\times d}{\boldsymbol{\varphi}(N)^\top} \end{bmatrix} \qquad \underset{N\times p}{E} = \begin{bmatrix} \underset{1\times p}{\boldsymbol{e}(1)^\top} \\ \vdots \\ \underset{1\times p}{\boldsymbol{e}(N)^\top} \end{bmatrix}$$

$$\underset{d\times p}{\boldsymbol{\theta}'} = \left[\underbrace{A_1 \cdots A_{n_a}}_{p\times(n_a\cdot p)} \; \underbrace{B_1 \cdots B_{n_b}}_{p\times(n_b\cdot p)}\right]^\top$$

Then the linear regression can be rewritten as: $\underset{N\times p}{Y} = \underset{N\times d}{\Phi} \cdot \underset{d\times p}{\boldsymbol{\theta}'} + \underset{N\times p}{E}$

and the **LS estimate** is given by:

$$\underset{d\times p}{\widehat{\boldsymbol{\theta}}'_N} = \underset{d\times N}{(\Phi^\top\Phi)^{-1}}\underset{d\times N}{\Phi^\top}\underset{N\times p}{Y}$$

Example: MIMO Least Squares estimation

Consider the following **deterministic** system with $m = 2$ inputs and $p = 2$ outputs:

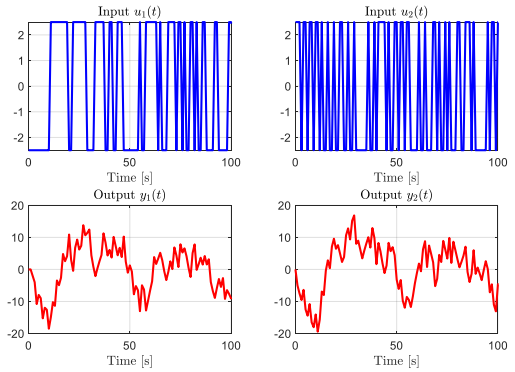
$$\mathcal{S}: \begin{cases} y_1(t) = 0.5y_1(t-1) + 0.4y_2(t-1) + u_1(t-1) + u_2(t-1) \\ y_2(t) = 0.4y_1(t-1) + 0.5y_2(t-1) + u_1(t-1) - u_2(t-1) \end{cases}$$

$$\mathcal{S}: \boldsymbol{y}(t) = A_1\boldsymbol{y}(t-1) + B_1\boldsymbol{u}(t-1)$$

$$\underset{\substack{p \times p \\ 2 \times 2}}{A_1} = \begin{bmatrix} 0.5 & 0.4 \\ 0.4 & 0.5 \end{bmatrix} \qquad \underset{\substack{p \times m_u \\ 2 \times 2}}{B_0} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \qquad \underset{\substack{p \times 1 \\ 2 \times 1}}{\boldsymbol{y}(t)} = \begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix} \qquad \underset{\substack{m_u \times 1 \\ 2 \times 1}}{\boldsymbol{u}(t)} = \begin{bmatrix} u_1(t) \\ u_2(t) \end{bmatrix}$$

Example: MIMO Least Squares estimation

Simulate the system using PRBS inputs with amplitudes $[-2.5, +2.5]$



By setting $\underset{1 \times 4}{\boldsymbol{\varphi}^\top(t)} = [\boldsymbol{y}(t-1)^\top \ \boldsymbol{u}(t-1)^\top]$ and constructing Y and Φ we get:

$$\underset{\substack{d \times p \\ 4 \times 2}}{\widehat{\boldsymbol{\theta}'}_N} = (\Phi^\top\Phi)^{-1}\Phi^\top Y = \begin{matrix} A_1 \\ \begin{bmatrix} 0.5 & 0.4 \\ 0.4 & 0.5 \end{bmatrix} \\ \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \\ B_0 \end{matrix}$$

Outline

Parameters estimation with Extended Kalman Filter

Consider a simple model for the mechanical part of a DC motor

$$J\dot{\omega}(t) = k_t I(t) - c_1\omega(t) \quad \Rightarrow \quad \dot{\omega}(t) = \frac{\tau_m(t) - c_1\omega(t)}{J}$$

- Armature current: $I(t)$
- Motor speed: $\omega(t)$
- Viscous friction coefficient: c_1
- Motor inertia: J

Discretize the system with $\dot{\omega}(t) \approx \frac{\omega(t+1)-\omega(t)}{T_s}$ where T_s is the sampling time, and assume that motor speed and acceleration are measurable outputs, so that

$$\begin{cases} \omega(t+1) = \omega(t) + \frac{T_s}{J}\tau_m(t) - c_1 \cdot \omega(t)\frac{T_s}{J} \\ y_1(t) = \omega(t) \\ y_2 = \dot{\omega}(t) = \big(\tau_m(t) - c_1\omega(t)\big)/J \end{cases}$$

Advanced material

Parameters estimation with Extended Kalman Filter

To estimate c_1, add a new state to express its time evolution

$$\begin{cases} \omega(t+1) = \omega(t) + \frac{T_s}{J}\tau_m(t) - c_1 \cdot \omega(t)\frac{T_s}{J} + v_1^1(t) \\ c_1(t+1) = c(t) + v_\theta(t) \\ y_1(t) = \omega(t) + v_2^1(t) \\ y_2 = -\frac{c_1}{J}\omega(t) + \frac{1}{J}\tau_m(t) + v_2^2(t) \end{cases}$$

$$\boldsymbol{x}(t) \coloneqq \begin{bmatrix} \omega(t) \\ c_1(t) \end{bmatrix} \qquad \boldsymbol{y}(t) \coloneqq \begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix}$$

$$\boldsymbol{v}_1(t) \coloneqq \begin{bmatrix} v_1^1(t) \\ v_\theta(t) \end{bmatrix} \sim \mathrm{WN}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} V_1 & 0 \\ 0 & V_\theta \end{bmatrix}\right)$$

$$\boldsymbol{v}_2(t) \coloneqq \begin{bmatrix} v_2^1(t) \\ v_2^1(t) \end{bmatrix} \sim \mathrm{WN}(0, V_2)$$

The system became **nonlinear** due to the multiplication of the new state $c_1(t)$ with $\omega(t)$

Parameters estimation with Extended Kalman Filter

The nonlinear system can be written as

$$\begin{cases} \boldsymbol{x}(t+1) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t)) + \boldsymbol{v}_1(t) \\ \boldsymbol{y}(t) = \boldsymbol{h}(\boldsymbol{x}(t), \boldsymbol{u}(t)) + \boldsymbol{v}_2(t) \end{cases}$$

$$\boldsymbol{x}(t) \coloneqq \begin{bmatrix} \omega(t) \\ c_1(t) \end{bmatrix} \qquad \boldsymbol{y}(t) \coloneqq \begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix} \qquad \boldsymbol{v}_2(t) \coloneqq \begin{bmatrix} v_2^1(t) \\ v_2^1(t) \end{bmatrix} \sim \mathrm{WN}(0, V_2)$$

$$\boldsymbol{v}_1(t) \coloneqq \begin{bmatrix} v_1^1(t) \\ v_\theta(t) \end{bmatrix} \sim \mathrm{WN}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} V_1 & 0 \\ 0 & V_\theta \end{bmatrix} = V_{1\theta}\right)$$

The EKF recursion are similar to those of the KF, except that, at each time instant, the **Jacobian** matrices approximating the functions $f(\cdot)$ and $h(\cdot)$ have to be computed

$$\boldsymbol{F}(t) = \frac{\partial \boldsymbol{f}(\cdot)}{\partial \boldsymbol{x}}\Big|_{\hat{\boldsymbol{x}}(t-1|t-1),\boldsymbol{u}(t)} \qquad \boldsymbol{H}(t) = \frac{\partial \boldsymbol{h}(\cdot)}{\partial \boldsymbol{x}}\Big|_{\hat{\boldsymbol{x}}(t|t-1),\boldsymbol{u}(t)}$$

Parameters estimation with Extended Kalman Filter

Extended Kalman Filter (EKF)

$$\begin{cases} \hat{\boldsymbol{x}}(t|t-1) = \boldsymbol{f}(\hat{\boldsymbol{x}}(t-1|t-1), \boldsymbol{u}(t)) \\ \hat{\boldsymbol{x}}(t|t) = \hat{\boldsymbol{x}}(t|t-1) + \tilde{K}_0(t)\boldsymbol{\varepsilon}_y(t) \\ \hat{\boldsymbol{y}}(t|t-1) = \boldsymbol{h}(\hat{\boldsymbol{x}}(t|t-1), \boldsymbol{u}(t)) \end{cases}$$

$$\begin{cases} \hat{\boldsymbol{x}}(1|0) = \boldsymbol{x}^0 \\ P_0(1|0) = P_0^0 \end{cases}$$

$P_0(t|t)$ is the covariance matrix of the filtering estimate $\hat{\boldsymbol{x}}(t|t)$

$$\begin{cases} \boldsymbol{\varepsilon}_y(t) = \boldsymbol{y}(t) - \hat{\boldsymbol{y}}(t|t-1) \\ \tilde{K}_0(t) = P_0(t|t-1)\boldsymbol{H}^\top(t)\tilde{\mathrm{E}}(t)^{-1} \\ \tilde{\mathrm{E}}(t) = \boldsymbol{H}(t)P_0(t|t-1)\boldsymbol{H}^\top(t) + V_2 \\ P_0(t|t-1) = \boldsymbol{F}(t)P_0(t-1|t-1)\boldsymbol{F}^\top(t) + V_{1\theta} \\ P_0(t|t) = \left(I_n - \tilde{K}_0(t)\boldsymbol{H}(t)\right)P_0(t|t-1) \end{cases}$$

$$\boldsymbol{F}(t) = \frac{\partial \boldsymbol{f}(\cdot)}{\partial \boldsymbol{x}}\Big|_{\hat{\boldsymbol{x}}(t-1|t-1),\boldsymbol{u}(t)}$$

$$\boldsymbol{H}(t) = \frac{\partial \boldsymbol{h}(\cdot)}{\partial \boldsymbol{x}}\Big|_{\hat{\boldsymbol{x}}(t|t-1),\boldsymbol{u}(t)}$$

Parameters estimation with Extended Kalman Filter

In our example we have

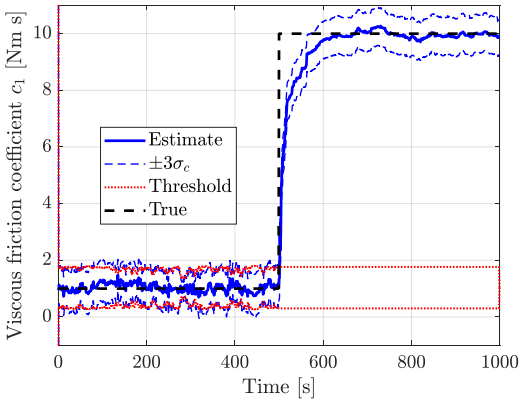
$$\boldsymbol{F}(t) = \frac{\partial \boldsymbol{f}(\cdot)}{\partial \boldsymbol{x}} = \begin{bmatrix} \frac{\partial f_1(\cdot)}{\partial x_1} & \frac{\partial f_1(\cdot)}{\partial x_2} \\ \frac{\partial f_2(\cdot)}{\partial x_1} & \frac{\partial f_2(\cdot)}{\partial x_2} \end{bmatrix} = \begin{bmatrix} 1 - \frac{T_s}{J}c(t) & -\frac{T_s}{J}\omega(t) \\ 0 & 1 \end{bmatrix}$$

$$\boldsymbol{H}(t) = \frac{\partial \boldsymbol{h}(\cdot)}{\partial \boldsymbol{x}} = \begin{bmatrix} \frac{\partial h_1(\cdot)}{\partial x_1} & \frac{\partial h_1(\cdot)}{\partial x_2} \\ \frac{\partial h_2(\cdot)}{\partial x_1} & \frac{\partial h_2(\cdot)}{\partial x_2} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ -\frac{c(t)}{J} & -\frac{\omega(t)}{J} \end{bmatrix}$$

Parameters estimation with Extended Kalman Filter

- $u(t) \sim \mathrm{WN}(0,1)$
- $V_{1\theta} = \begin{bmatrix} V_1 = 10^{-6} & 0 \\ 0 & V_\theta = 0.05 \end{bmatrix}$
- $V_2 = \begin{bmatrix} 10^{-4} & 0 \\ 0 & 10^{-4} \end{bmatrix}$
- $P_0(0) = \begin{bmatrix} 1 & 0 \\ 0 & 100 \end{bmatrix}$
- $T_s = 1$
- $J = 10$
- $c_1 = \begin{cases} 1 & \text{if } t \leq 500 \\ 10 & \text{if } t > 500 \end{cases}$

Thresholds are set to $\eta = \pm 3 \cdot \mathrm{mean}(\sigma_c)$ where σ_c is the estimation variance of c_1 during healthy state



Setup of the model variances for EKF estimation

- V_2: start from the analysis of the **uncertainty** of the **observations**. If these are the result of measurements carried out with special instruments, V_2 is given from the variance of measurement errors
- V_1: it is chosen to get at a reasonable **tradeoff** between the «trust» given to observations with respect to that provided by the model and the initial conditions
- V_θ: similar to V_1. A **greater** V_θ allows a **more rapid change** (but also more vaiability) of the parameter estimate (since more trust is given to new information with respect to the model and the initial condition)

Outline

Parameters estimation for continuous-time systems

Parameters estimation is particularly useful when there is a continuous-time model of the system, so that changes in **model parameters** can be related to **process coefficients**

For instance, consider a DC motor with:

Parameter	Value
Power	$P = 550$ W
Nominal speed	$\bar{\omega} = 2500$ rpm
Armature resistance	$R = 1.52\ \Omega$
Armature inductance	$L = 6.82 \cdot 10^{-3}\ \Omega \cdot \mathrm{s}$
Magnetic flux	$k_t = 0.33$ Vs
Motor inertia	$J = 1.92 \cdot 10^{-3}\ \mathrm{kg} \cdot \mathrm{m}^2$
Viscous friction	$c_1 = 0.36 \cdot 10^{-3}$ Nm · s
Coulomb friction	$c_0 = 0.11$ Nm

Measured signals:

- Armature voltage: $V(t)$
- Armature current: $I(t)$
- Motor speed: $\omega(t)$

Parameters estimation for continuous-time systems

The process equations, considering null opposing load, are:

$$\begin{cases} L\dot{I}(t) = -RI(t) - k_t\omega(t) + V(t) \\ J\dot{\omega}(t) = k_t I(t) - c_1\omega(t) - \boxed{c_0 \text{sign}(\omega(t))} \end{cases} \Rightarrow \tau_l(t) \qquad \begin{cases} L\dot{I}(t) = -RI(t) - k_t\omega(t) + V(t) \\ J\dot{\omega}(t) = k_t I(t) - c_1\omega(t) - \tau_l(t) \end{cases}$$

Assuming c_0 known, the model parameters $\boldsymbol{\theta}$ are related to the process coefficients as

$$\begin{cases} \dot{I}(t) = -\theta_1 I(t) - \theta_2\omega(t) + \theta_3 V(t) \\ \dot{\omega}(t) = \theta_4 I(t) - \theta_5\omega(t) - \theta_6\tau_l(t) \end{cases}$$

$$R = \frac{\theta_1}{\theta_3} \quad L = \frac{1}{\theta_3} \quad k_t = \frac{\theta_2}{\theta_3}$$

$$k_t = \frac{\theta_4}{\theta_6} \quad J = \frac{1}{\theta_6} \quad c_1 = \frac{\theta_5}{\theta_6}$$

Parameters estimation for continuous-time systems

Once model parameters $\boldsymbol{\theta} = [\theta_1 \ \theta_2 \ \dots \ \theta_6]^\top$ have been estimated, the computation of process coefficients $\boldsymbol{\pi} = [R \ \ L \ \ k_t \ J \ \ c_1]^\top$ is possible

The estimates $\hat{\boldsymbol{\theta}}(t)$ can be computed using **two sets of RLS** models. However, **signal derivatives** are necessary

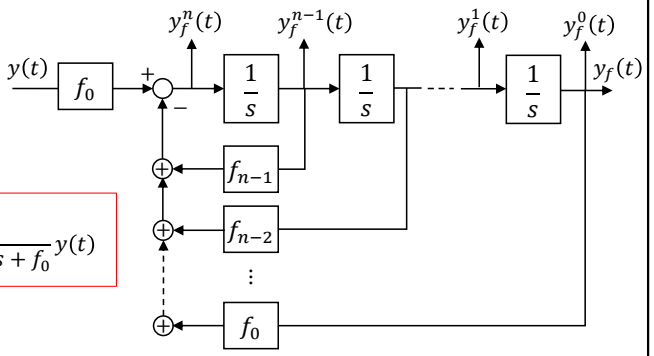
Fault detection is performed by looking at changes in the $\boldsymbol{\theta}$ or $\boldsymbol{\pi}$ parameters **singularly**. Fault isolation might be possible considering chages in $\boldsymbol{\theta}$ or $\boldsymbol{\pi}$ **toghether,** to look for patterns

Signal derivatives can be computed using **state-variable filters** designed as **low-pass Butterworth filter**

State-variable filters

A state-variable filter of order n provides the first $n-1$ **(low-pass filtered) derivatives** of the input signal $y(t)$

$$y_f(t) = \frac{f_0}{s^n + f_{n-1}s^{n-1} + \cdots + f_1 s + f_0} y(t)$$



State-variable filters

1. Design a **Butterworth filter** of order n and cut-off frequency ω_c using software tools. Butterworth filters are **maximally flat** within the passband of the filter

$$y_f(t) = \frac{f_0}{s^n + f_{n-1}s^{n-1} + \cdots + f_1 s + f_0} y(t)$$ **Low-pass filter**

2. Convert the filter into **state-space controllable form**

$$\begin{cases} \dot{\boldsymbol{x}}(t) = A_c\boldsymbol{x}(t) + B_c y(t) \\ y_f(t) = C_c\boldsymbol{x}(t) \end{cases} \quad A_c = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ \vdots & \ddots & 1 & \ddots & \vdots \\ \vdots & & \ddots & \ddots & 0 \\ 0 & \cdots & \cdots & 0 & 1 \\ -f_0 & f_1 & \cdots & \cdots & f_{n-1} \end{bmatrix} \quad B_c = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ f_0 \end{bmatrix} \quad C_c = [1 \ \ 0 \ \ \cdots \ \ 0]$$

State-variable filters

3. Get desired **low-pass filtered derivatives** from the **state variables** of the filter, as

$$\begin{cases} \dot{\boldsymbol{x}}(t) = A_c \boldsymbol{x}(t) + B_c y(t) \\ y_f(t) = C_c \boldsymbol{x}(t) \end{cases}$$

$$\boldsymbol{x}(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \\ \vdots \\ x_n(t) \end{bmatrix} = \begin{bmatrix} y_f(t) \\ y_f^1(t) \\ \vdots \\ y_f^{n-1}(t) \end{bmatrix}$$

Notice that the first state variable $x_1(t)$ is equal to a low-pass filtered version of the input signal $y(t)$, that is $x_1(t) = y_f(t)$

Example: state-variable filter

Compute the first 2 derivatives of the noisy sinusoidal signal

$$y(t) = \sin(2 \cdot t) + e(t) \qquad e(t) \sim \mathrm{WN}(0, \lambda^2)$$

Where λ^2 is such that $\mathrm{SNR} = \frac{\mathrm{var}[\sin(2 \cdot t)]}{\mathrm{var}[e(t)]} = 50$

1. Design a Butterworth filter of order $n = 3$ with cut-off frequency of $\omega_c = 5$ rad/s

```
butter(n, wc, 'low', 's')
```

$$y_f(t) = \frac{125}{s^3 + 10s^2 + 50s + 125} y(t)$$

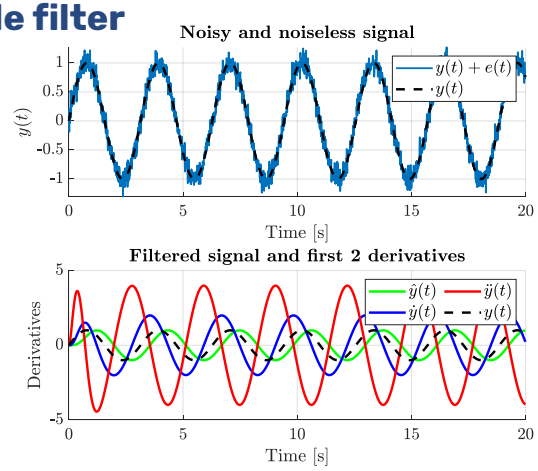
Example: state-variable filter

2. Convert into state-space controllable canonical form

$$A_c = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -125 & -50 & -10 \end{bmatrix}$$

$$B_c = \begin{bmatrix} \\ \\ 125 \end{bmatrix}$$

$$C_c = [1 \quad 0 \quad 0]$$



UNIVERSITÀ DEGLI STUDI DI BERGAMO
Dipartimento di Ingegneria Gestionale, dell'Informazione e della Produzione

UNIVERSITÀ DEGLI STUDI DI BERGAMO | Dipartimento di Ingegneria Gestionale, dell'Informazione e della Produzione

ADAPTIVE LEARNING, ESTIMATION AND SUPERVISION OF DYNAMICAL SYSTEMS (ALES)

Lecture 14: Robust fault detection

Master Degree in COMPUTER ENGINEERING

Data Science and Data Engineering Curriculum

SPEAKER
Prof. Mirko Mazzoleni

PLACE
University of Bergamo

Syllabus

1. **Recursive and adaptive identification**
   - 1.1 Recursive ARX estimation (RLS)
   - 1.2 Least Mean Squares (LMS)
   - 1.3 Instrumental Variables (IV)
2. **Closed-loop identification**
3. **Subspace and MIMO identification**
   - 3.1 Singular Value Decomposition
   - 3.2 Impulse data: Ho-Kalman, Kung algorithms
   - 3.3 Generic I/O data: the MOESP algorithm
4. **Supervision of dynamical systems**
   - 4.1 Introduction to fault diagnosis
   - 4.2 Model-based fault diagnosis
   - 4.3 Parity space approaches
   - 4.4 Observer-based approaches
   - 4.5 Signal-based fault diagnosis
   - 4.6 Knowledge-based fault diagnosis

CASE STUDIES

- Virtual Reference Feedback Tuning
- Nuclear particles classification
- Leak detection in an industrial valve
- Bearing fault identification

Outline

1. Robust fault detection in the parity-space framework
2. Optimally robust fault detection in the parity-space framework
3. Robust fault detection in the deterministic observer framework: UIO

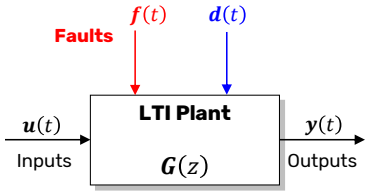
Outline

1. **Robust fault detection in the parity-space framework**
2. Optimally robust fault detection in the parity-space framework
3. Robust fault detection in the deterministic observer framework: UIO

Robust parity-space residual generators

Consider a LTI dynamical systems with **faults** and **disturbances**

$$\begin{cases} \underset{n\times 1}{\boldsymbol{x}(t+1)} = \underset{n\times n}{A}\cdot\underset{n\times 1}{\boldsymbol{x}(t)} + \underset{n\times m_u}{B}\cdot\underset{m_u\times 1}{\boldsymbol{u}(t)} + \underset{n\times m_f}{B_f}\cdot\underset{m_f\times 1}{\boldsymbol{f}(t)} + \underset{n\times m_d}{B_d}\cdot\underset{m_d\times 1}{\boldsymbol{d}(t)} \\ \underset{p\times 1}{\boldsymbol{y}(t)} = \underset{p\times n}{C}\cdot\underset{n\times 1}{\boldsymbol{x}(t)} + \underset{p\times m_u}{D}\cdot\underset{m_u\times 1}{\boldsymbol{u}(t)} + \underset{p\times m_f}{D_f}\cdot\underset{m_f\times 1}{\boldsymbol{f}(t)} + \underset{p\times m_d}{D_d}\cdot\underset{m_d\times 1}{\boldsymbol{d}(t)} \end{cases}$$



- $\boldsymbol{d}(t) \in \mathbb{R}^{m_d\times 1}$: additive disturbance signals
- $\boldsymbol{f}(t) \in \mathbb{R}^{m_f\times 1}$: additive fault signals
- We assume no noise signals $\boldsymbol{w}(t)$ are present

The aim is to design a parity-space residual generator so that the residuals are **decoupled** (not influenced) from the **disturbaces** (so to avoid false alarms)

Robust parity-space residual generators

Suppose that both **disturbances** and **faults** affect the system, i.e. $\boldsymbol{d}(t) \neq 0$ and $\boldsymbol{f}(t) \neq 0$.

Define the quantities:

$$\underset{m_d(s+1)\times 1}{\overleftarrow{\boldsymbol{d}_{s+1}}(t)} = \begin{bmatrix} \underset{m_d\times 1}{\boldsymbol{d}(t-s)} \\ \boldsymbol{d}(t-s+1) \\ \vdots \\ \boldsymbol{d}(t) \end{bmatrix} \qquad \underset{p(s+1)\times m_d(s+1)}{\mathcal{D}_{s+1}} = \begin{bmatrix} D_d & 0 & \cdots & \cdots & 0 \\ \underset{p\times m_d}{CB_d} & \ddots & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ CA^{s-1}B_d & \cdots & \cdots & CB_d & D_d \end{bmatrix}$$

Then, the parity relations (with faults and disturbances) become

$$\overleftarrow{\boldsymbol{y}_{s+1}}(t) = \mathcal{O}_{s+1}\cdot\boldsymbol{x}(t-s) + \mathcal{T}_{s+1}\cdot\overleftarrow{\boldsymbol{u}_{s+1}}(t) + \mathcal{D}_{s+1}\cdot\overleftarrow{\boldsymbol{d}_{s+1}}(t) + \mathcal{F}_{s+1}\cdot\overleftarrow{\boldsymbol{f}_{s+1}}(t)$$

Robust parity-space residual generators

The **effects** of the **faults** and **disturbances** on the **residuals** are given by

$$\boldsymbol{r}(t) = V_{s+1}\left(\overleftarrow{\boldsymbol{y}_{s+1}}(t) - \mathcal{T}_{s+1}\overleftarrow{\boldsymbol{u}_{s+1}}(t)\right) = V_{s+1}\left(\mathcal{O}_{s+1}\boldsymbol{x}(t-s) + \mathcal{D}_{s+1}\overleftarrow{\boldsymbol{d}_{s+1}}(t) + \mathcal{F}_{s+1}\overleftarrow{\boldsymbol{f}_{s+1}}(t)\right)$$

$$\underset{q\times 1}{\boldsymbol{r}(t)} = \overset{q\times p(s+1)}{V_{s+1}}\cdot\left(\underset{p(s+1)\times m_d(s+1)}{\mathcal{D}_{s+1}}\cdot\overset{m_d(s+1)\times 1}{\overleftarrow{\boldsymbol{d}_{s+1}}(t)} + \underset{p(s+1)\times m_f(s+1)}{\mathcal{F}_{s+1}}\cdot\overset{m_f(s+1)\times 1}{\overleftarrow{\boldsymbol{f}_{s+1}}(t)}\right)$$

Residuals generator (internal form)

If it holds that $V_{s+1}\mathcal{D}_{s+1} = \mathbf{0}$ **and** $V_{s+1}\mathcal{F}_{s+1} \neq \mathbf{0}$, the residual is **not zero** when a fault is present, and so **robust fault detection** is achieved, i.e. the residuals depend only on faults and not on disturbances

Robust parity-space residual generators

Thus, the rows $\boldsymbol{v}_{s+1}^{\top} \in \mathbb{R}^{1\times p(s+1)}$ of parity matrix V_{s+1} (of dimensions $q \times p(s+1)$) should belong to the **left nullspace** of the matrix

$$\underset{p(s+1)\times(n+m_d(s+1))}{\Gamma_{s+1}} \coloneqq \begin{bmatrix} \underset{p(s+1)\times n}{\mathcal{O}_{s+1}} & \underset{p(s+1)\times m_d(s+1)}{\mathcal{D}_{s+1}} \end{bmatrix}$$

For this V_{s+1} to exist, it is required that the left nullspace of Γ_{s+1} is different from the empty set, i.e.

$$p(s+1) - \operatorname{rank}(\Gamma_{s+1}) > 0$$

This is **generally** true if $p > m_d$

Robust parity-space residual generators

For faults to be detectable, the **fault vector $f(t)$ must not belong** to $\mathcal{C}(\mathcal{D}_{s+1})$, otherwise the matrix V_{s+1} nullifies also the **faults** in addition to the **disturbances**

This holds if

$$\text{rank}([\mathcal{O}_{s+1} \quad \mathcal{D}_{s+1} \quad \mathcal{F}_{s+1}]) > \text{rank}([\mathcal{O}_{s+1} \quad \mathcal{D}_{s+1}])$$

i.e. the columns of the matrix $\mathcal{F}_{s+1}$ **are not linear combinations** of the columns of the matrix $[\mathcal{O}_{s+1} \quad \mathcal{D}_{s+1}]$

Example: robust parity-space residual generation

Consider the following MIMO system with $m_u = 2, p = 3, m_f = 1,$ sampled at $T_s = 1\,\text{s}$.

Assume there are $m_d = 3$ disturbances acting on states and outputs

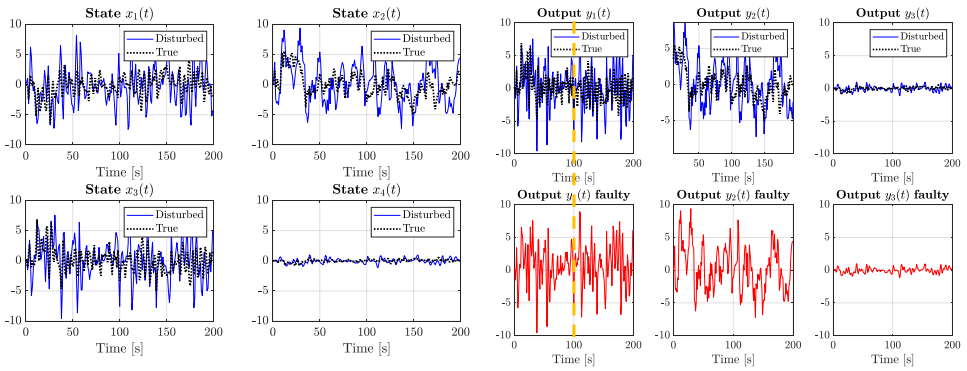
$$\underset{\substack{n\times n\\ 4\times 4}}{A} = \begin{bmatrix} 0.5 & -0.7 & 0.7 & 0 \\ 0 & 0.8 & 0.06 & 0 \\ -1 & 0 & 0 & 0.1 \\ 0 & 0 & -0.1 & 0.4 \end{bmatrix} \quad \underset{\substack{n\times m_u\\ 4\times 2}}{B} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \quad \underset{\substack{p\times n\\ 3\times 4}}{C} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \underset{\substack{p\times m_u\\ 3\times 2}}{D} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

$$\underset{\substack{n\times m_f\\ 4\times 1}}{B_f} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \underset{\substack{p\times m_f\\ 3\times 1}}{D_f} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \quad \underset{\substack{n\times m_d\\ 4\times 3}}{B_d} = \begin{bmatrix} 0 & 1 & 0 \\ 2 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad \underset{\substack{p\times m_d\\ 3\times 3}}{D_d} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

- $\boldsymbol{d}(t) \sim \text{WN}(\mathbf{0}, I_3)$
- $\boldsymbol{u}(t) \sim \text{WN}(\mathbf{0}, I_2)$

Collect $N = 200$ data. A step-like fault of amplitude 1 acts on the system at time $t = 100$ s

Example: robust parity-space residual generation

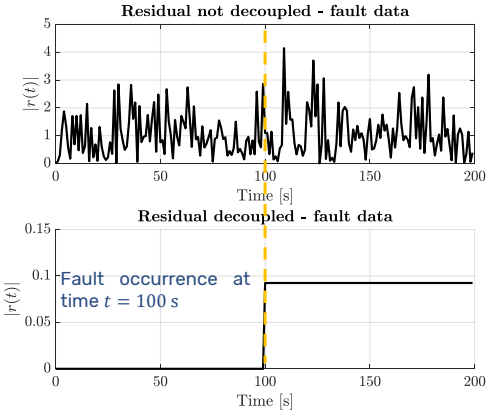


Example: robust parity-space residual generation

A single parity vector is chosen as $\boldsymbol{v}_{s+1}^{\top} = V_{s+1}(1,:)$

The non robust approach generates a residual which is submerged by the disturbance

The robust approach decouples the residual from the disturbance and it is able to detect the fault



Outline

1. Robust fault detection in the parity-space framework
2. **Optimally robust fault detection in the parity-space framework**
3. Robust fault detection in the deterministic observer framework: UIO

Optimally robust parity space residual generators

Consider the case where we want a tradeoff between:

1) Maximal robustness *(minimal sensitivity)* to **disturbances**

2) Maximal sensitivity to **faults**

The situation is especially important where **not all disturbances** can be decoupled

In general, the vectors in $v_{s+1}^{\top}$ that satisfy **1)** may not satisfy also **2)**. A widely used index, to be **minimized**, is

$$J(\boldsymbol{v}_{s+1}) = \frac{\boldsymbol{v}_{s+1}^{\top}\Gamma_{s+1}\boldsymbol{v}_{s+1}}{\boldsymbol{v}_{s+1}^{\top}\mathcal{F}_{s+1}\boldsymbol{v}_{s+1}}$$

- $\Gamma_{s+1} \coloneqq [\mathcal{O}_{s+1} \quad \mathcal{D}_{s+1}]$

Advanced material

Optimally robust parity space residual generators

Definition

Let A, B two $m \times n$ matrices. Given two quadratic forms $J_1 = \boldsymbol{x}^{\top}AA^{\top}\boldsymbol{x} \quad J_2 = \boldsymbol{x}^{\top}BB^{\top}\boldsymbol{x}$

The equation:

$$\det(AA^{\top} - \lambda \cdot BB^{\top})$$

is called the characteristic equation of the **regular matrix pencil** $\boldsymbol{x}^{\top}AA^{\top}\boldsymbol{x} - \lambda \cdot \boldsymbol{x}^{\top}BB^{\top}\boldsymbol{x}$

The roots of this characteristic equations are the **generalized eigenvalues** of the pencil:

$$\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_m$$

The **generalized eigenvectors** are vectors $\boldsymbol{w}_i$ of dimensions $m \times 1$ such that

$$(AA^{\top} - \lambda_i \cdot BB^{\top})\boldsymbol{w}_i = \boldsymbol{0}$$

Advanced material

Optimally robust parity space residual generators

It can be shown that the vector $\boldsymbol{v}_{s+1}^{*}$ that minimizes

$$J(\boldsymbol{v}_{s+1}) = \frac{\boldsymbol{v}_{s+1}^{\top}\Gamma_{s+1}\boldsymbol{v}_{s+1}}{\boldsymbol{v}_{s+1}^{\top}\mathcal{F}_{s+1}\boldsymbol{v}_{s+1}}$$

is such that

$$\boldsymbol{v}_{s+1}^{*} = \boldsymbol{w}_1 \qquad J(\boldsymbol{v}_{s+1}^{*}) = \lambda_1$$

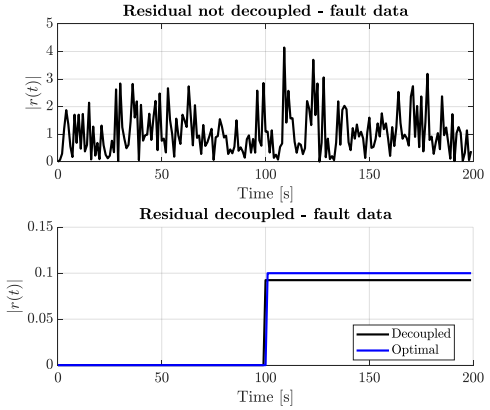
where:

- $\boldsymbol{w}_1$ is the first **eigenvector** of the matrix pencil $\boldsymbol{v}_{s+1}^{\top}\Gamma_{s+1}\Gamma_{s+1}^{\top}\boldsymbol{v}_{s+1} - \lambda \cdot \boldsymbol{v}_{s+1}^{\top}\mathcal{F}_{s+1}\mathcal{F}_{s+1}^{\top}\boldsymbol{v}_{s+1}$
- λ_1 its corresponding **eigenvalue**

Advanced material

Example: robust parity-space residual generation

Considering the same example as before, the optimal parity vector has a bit **more sensitivity** to the fault



Advanced material

Outline

1. Robust fault detection in the parity-space framework
2. Optimally robust fault detection in the parity-space framework
3. **Robust fault detection in the deterministic observer framework: UIO**

Theory of Unknown Input Observers (UIO)

Consider a LTI dynamical system

$$\begin{cases} \underset{n\times 1}{\boldsymbol{x}(t+1)} = \underset{n\times n}{A}\cdot\underset{n\times 1}{\boldsymbol{x}(t)} + \underset{n\times m_u}{B}\cdot\underset{m_u\times 1}{\boldsymbol{u}(t)} + \underset{n\times m_d}{B_d}\cdot\underset{m_d\times 1}{\boldsymbol{d}(t)} \\ \underset{p\times 1}{\boldsymbol{y}(t)} = \underset{p\times n}{C}\cdot\underset{n\times 1}{\boldsymbol{x}(t)} \end{cases}$$

- $\boldsymbol{x} \in \mathbb{R}^{n\times 1}$ is the **state** vector
- $\boldsymbol{u} \in \mathbb{R}^{m_u\times 1}$ is the **input** vector
- $\boldsymbol{d} \in \mathbb{R}^{m_d\times 1}$ is the **disturbance** vector
- $\boldsymbol{y} \in \mathbb{R}^{p\times 1}$ is the **output** vector

Aim: find an **estimate** $\hat{\boldsymbol{x}}(t)$ of $\boldsymbol{x}(t)$, using $\boldsymbol{u}(t)$ and $\boldsymbol{y}(t)$, despite the presence of $\boldsymbol{d}(t)$

Remarks:

- We assume $B_d \in \mathbb{R}^{n\times m_d}$ *full column rank*. If not, we can write $B_d\boldsymbol{d}(t) = B_d'B_d''\boldsymbol{d}(t)$, e.g. by a QR decomposition, and use B_d' as new disturbance matrix, where B_d' is square and orthonormal (and thus full rank)

Preliminaries and assumptions

Remarks:

- The term $B_d\boldsymbol{d}(t)$ can be used to describe additive **disturbances**, interconnecting terms in large scale systems, **nonlinear terms** in system dynamics, terms arising from **time-varying** system dynamics, **linearization** and model reduction errors, **parameter** variations
- Some systems present an input term $D\boldsymbol{u}(t)$ at the output, such that

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

Since $D\boldsymbol{u}(t)$ is known, we can **define a new output** $\bar{\boldsymbol{y}}(t)$ as

$$\bar{\boldsymbol{y}}(t) = \boldsymbol{y}(t) - D\boldsymbol{u}(t) = C\boldsymbol{x}(t)$$

Preliminaries and assumptions

- Let $D_d\boldsymbol{d}(t)$ such that $\boldsymbol{y}(t) = C\boldsymbol{x}(t) + D_d\boldsymbol{d}(t)$, with $D_d \in \mathbb{R}^{p\times m_d}$, and assume that $p > m_d$ (**more outputs than disturbances**)

If the outputs are **independent**, the output term $D_d\boldsymbol{d}(t)$ can be nulled by computing the $(p - m_d) \times p$ matrix $T_y \in \mathcal{N}(D_d^\top)$

$$\underset{(p-m_d)\times p}{T_y} \cdot \underset{p\times m_d}{D_d} = \underset{(p-m_d)\times m_d}{\mathbf{0}}$$

Then, we can replace $\boldsymbol{y}(t)$ with $\boldsymbol{y}_D(t)$ and C with T_yC to get the alternative system

$$\begin{cases} \underset{n\times 1}{\boldsymbol{x}(t+1)} = \underset{n\times n}{A} \cdot \underset{n\times 1}{\boldsymbol{x}(t)} + \underset{n\times m_u}{B} \cdot \underset{m_u\times 1}{\boldsymbol{u}(t)} + \underset{n\times m_d}{B_d} \cdot \underset{m_d\times 1}{\boldsymbol{d}(t)} \\ \underset{(p-m_d)\times p}{\boldsymbol{y}_D(t)} = \underset{(p-m_d)\times p}{T_yC} \cdot \underset{n\times 1}{\boldsymbol{x}(t)} \end{cases}$$

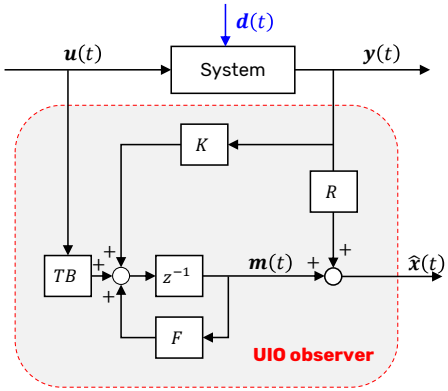
Theory of Unknown Input Observers (UIO)

The structure of a **full-order UIO** is described as:

$$\begin{cases} \underset{n\times 1}{\boldsymbol{m}(t+1)} = \underset{n\times n}{F} \cdot \underset{n\times 1}{\boldsymbol{m}(t)} + \underset{n\times p}{K} \cdot \underset{p\times 1}{\boldsymbol{y}(t)} + \underset{n\times n}{TB} \cdot \underset{m_u\times 1}{\boldsymbol{u}(t)} \\ \underset{n\times 1}{\hat{\boldsymbol{x}}(t)} = \underset{n\times 1}{\boldsymbol{m}(t)} + \underset{n\times p}{R} \cdot \underset{p\times 1}{\boldsymbol{y}(t)} \end{cases}$$

where $K = K_1 + K_2$

- The output of the UIO is an estimate $\hat{\boldsymbol{x}}(t)$ of the system state $\boldsymbol{x}(t)$
- The full-order UIO allows to design **directional** and **minimum variance** residuals



Theory of Unknown Input Observers (UIO)

Let's apply the UIO

$$\begin{cases} \underset{n\times 1}{\boldsymbol{m}(t+1)} = \underset{n\times n}{F} \cdot \underset{n\times 1}{\boldsymbol{m}(t)} + \underset{n\times p}{K} \cdot \underset{p\times 1}{\boldsymbol{y}(t)} + \underset{n\times n}{TB} \cdot \underset{m_u\times 1}{\boldsymbol{u}(t)} \\ \underset{n\times 1}{\hat{\boldsymbol{x}}(t)} = \underset{n\times 1}{\boldsymbol{m}(t)} + \underset{n\times p}{R} \cdot \underset{p\times 1}{\boldsymbol{y}(t)} \end{cases}$$

to the system

$$\begin{cases} \underset{n\times 1}{\boldsymbol{x}(t+1)} = \underset{n\times n}{A} \cdot \underset{n\times 1}{\boldsymbol{x}(t)} + \underset{n\times m_u}{B} \cdot \underset{m_u\times 1}{\boldsymbol{u}(t)} + \underset{n\times m_d}{B_d} \cdot \underset{m_d\times 1}{\boldsymbol{d}(t)} \\ \underset{p\times n}{\boldsymbol{y}(t)} = \underset{p\times n}{C} \cdot \underset{n\times 1}{\boldsymbol{x}(t)} \end{cases}$$

Theory of Unknown Input Observers (UIO)

The state estimation error $\boldsymbol{\varepsilon}_x(t+1) \coloneqq \boldsymbol{x}(t+1) - \hat{\boldsymbol{x}}(t+1)$ can be written as

$$\begin{aligned} \boldsymbol{\varepsilon}_x(t+1) = {} & [A - RCA - K_1C] \cdot \boldsymbol{\varepsilon}_x(t) + \\ & + [-F + (A - RCA - K_1C)] \cdot \boldsymbol{m}(t) \\ & + [-K_2 + (A - RCA - K_1C)R] \cdot \boldsymbol{y}(t) \\ & + [-T + (I_n - RC)]B \cdot \boldsymbol{u}(t) + \\ & + [I_n - RC]B_d \cdot \boldsymbol{d}(t) \end{aligned}$$

If it holds that...

1. $\underset{n\times n}{F} = A - RCA - K_1C$
2. $\underset{n\times p}{K_2} = \underset{n\times p}{FR}$
3. $\underset{n\times n}{T} = I_n - RC$
4. $[RC - I_n]B_d = \underset{n\times m_d}{\mathbf{0}}$

...then

$$\boldsymbol{\varepsilon}_x(t+1) = F \cdot \boldsymbol{\varepsilon}_x(t)$$

⇨ If F is **stable**, then $\boldsymbol{\varepsilon}_x(t) \to \mathbf{0}$ as $t \to +\infty$, **despite** the presence of $\boldsymbol{d}(t)$

Theory of Unknown Input Observers (UIO)

Consider the state estimation error $\boldsymbol{\varepsilon_x}(t+1)$

$$\boldsymbol{\varepsilon_x}(t+1) \coloneqq \boldsymbol{x}(t+1) - \hat{\boldsymbol{x}}(t+1)$$

$$\boldsymbol{\varepsilon_x}(t+1) = A\boldsymbol{x}(t) + B\boldsymbol{u}(t) + B_d\boldsymbol{d}(t) - \boldsymbol{m}(t+1) - R\boldsymbol{y}(t+1)$$

$$= A\boldsymbol{x}(t) + B\boldsymbol{u}(t) + B_d\boldsymbol{d}(t) - F\boldsymbol{m}(t) - TB\boldsymbol{u}(t) - K\boldsymbol{y}(t) - R\boldsymbol{y}(t+1)$$

$$R\boldsymbol{y}(t+1) = RC\boldsymbol{x}(t+1) = RC[A\boldsymbol{x}(t) + B\boldsymbol{u}(t) + B_d\boldsymbol{d}(t)]$$

$$= A\boldsymbol{x}(t) - RCA\boldsymbol{x}(t) + B\boldsymbol{u}(t) - RCB\boldsymbol{u}(t) + B_d\boldsymbol{d}(t) - RCB_d\boldsymbol{d}(t) - F\boldsymbol{m}(t) - TB\boldsymbol{u}(t) - K\boldsymbol{y}(t)$$

Theory of Unknown Input Observers (UIO)

$$\boldsymbol{\varepsilon_x}(t+1) = [A - RCA] \cdot \boldsymbol{x}(t) + [-T + (I - RC)]B\boldsymbol{u}(t) + [I - RC]B_d\boldsymbol{d}(t) - F\boldsymbol{m}(t) - K\boldsymbol{y}(t)$$

$$-K\boldsymbol{y}(t) = -(K_1 + K_2) \cdot C\boldsymbol{x}(t) = -K_1C\boldsymbol{x}(t) - K_2C\boldsymbol{x}(t) = -K_1C\boldsymbol{x}(t) - K_2\boldsymbol{y}(t)$$

$$= [A - RCA - K_1C] \cdot \boldsymbol{x}(t) + [-T + (I - RC)]B\boldsymbol{u}(t) + [I - RC]B_d\boldsymbol{d}(t) - F\boldsymbol{m}(t) - K_2\boldsymbol{y}(t)$$

$$= [A - RCA - K_1C] \cdot (\boldsymbol{\varepsilon_x}(t) + \hat{\boldsymbol{x}}(t)) + [-T + (I - RC)]B\boldsymbol{u}(t) + [I - RC]B_d\boldsymbol{d}(t) - F\boldsymbol{m}(t) - K_2\boldsymbol{y}(t)$$

$$= [A - RCA - K_1C] \cdot (\boldsymbol{\varepsilon_x}(t) + \boldsymbol{m}(t) + R\boldsymbol{y}(t)) + [-T + (I - RC)]B\boldsymbol{u}(t) + [I - RC]B_d\boldsymbol{d}(t) - F\boldsymbol{m}(t) - K_2\boldsymbol{y}(t)$$

Theory of Unknown Input Observers (UIO)

$$= [A - RCA - K_1C] \cdot (\boldsymbol{\varepsilon_x}(t) + \boldsymbol{m}(t) + R\boldsymbol{y}(t)) + [-T + (I - RC)]B\boldsymbol{u}(t) + [I - RC]B_d\boldsymbol{d}(t) - F\boldsymbol{m}(t) - K_2\boldsymbol{y}(t)$$

$$\boldsymbol{\varepsilon_x}(t+1) = [A - RCA - K_1C] \cdot \boldsymbol{\varepsilon_x}(t) + [-F + (A - RCA - K_1C)] \cdot \boldsymbol{m}(t) +$$

$$+[-T + (I - RC)]B\boldsymbol{u}(t) + [I - RC]B_d\boldsymbol{d}(t) + [-K_2 + (A - RCA - K_1C)R]\boldsymbol{y}(t)$$

Theory of Unknown Input Observers (UIO)

1. $A - RCA - K_1C = F$

Let $\underset{n \times n}{A_1} = A - RCA = (I_n - RC)A = TA$. Then we have that $A_1 - K_1C = F$, and K_1 can be set by **pole placement**, if (C, A_1) is **observable**

Actually, (C, A_1) has only to be **detectable**, but in this case pole placement cannot be used. An alternative strategy for this case is outlined in *[Chen, Patton] Robust model fault diagnosis for dynamical systems*

Theory of Unknown Input Observers (UIO)

4. $[RC - I]B_d = \mathbf{0}$

This equation is solvable **if and only if**

This means that we must have $p \geq m_d$

$$\underset{p \times m_d}{\text{rank}(CB_d)} = \underset{n \times m_d}{\text{rank}(B_d)}$$

B_d is assumed full column rank

And a **special solution** (not the only one!) is

$$\underset{n \times p}{R^*} = \underset{n \times m_d}{B_d} \cdot \underset{m_d \times p}{(CB_d)^\dagger} = B_d \cdot [(CB_d)^\top CB_d]^{-1}(CB_d)^\top$$

Theory of Unknown Input Observers (UIO)

Here we only prove **sufficiency** (see *[Chen, Patton] Robust model fault diagnosis for dynamical systems* for further details)

Since $\text{rank}(CB_d) = \text{rank}(B_d)$, then CB_d is full-column rank and a left inverse $(CB_d)^\dagger$ exists, such that $(CB_d)^\dagger CB_d = I_{m_d}$. Therefore, using R^*, we have

$$[R^*C - I]B_d = \mathbf{0} \implies R^*CB_d = B_d \implies \left[B_d \cdot (CB_d)^\dagger\right]CB_d = B_d \implies B_d = B_d$$

So that R^* is a solution for 4. $[RC - I]B_d = \mathbf{0}$

Advanced material

Theory of Unknown Input Observers (UIO)

Remarks:

- The **number of independent rows** of C must **not be less** than the **number of the independent columns** of B_d to satisfy $\text{rank}(CB_d) = \text{rank}(B_d)$, i.e. for **disturbance decoupling only**
- In other words, the **maximum number of disturbances** which can be decoupled **cannot be larger** than the **number of the independent measurements**
- For **both disturbance decoupling and fault detectability**, it is required that

$$p > m_d$$

where p denotes the number of **independent** outputs

Theory of Unknown Input Observers (UIO)

Remarks:

- The UIO

$$\begin{cases} \underset{n \times 1}{\boldsymbol{m}(t+1)} = \underset{n \times n}{F} \cdot \underset{n \times 1}{\boldsymbol{m}(t)} + \underset{n \times p}{K} \cdot \underset{p \times 1}{\boldsymbol{y}(t)} + \underset{n \times n}{TB} \cdot \underset{m_u \times 1}{\boldsymbol{u}(t)} \\ \underset{n \times 1}{\hat{\boldsymbol{x}}(t)} = \underset{n \times 1}{\boldsymbol{m}(t)} + \underset{n \times p}{R} \cdot \underset{p \times 1}{\boldsymbol{y}(t)} \end{cases}$$

when $B_d = 0$, is **equivalent to a Luenberger observer** if we set $T = I_n$ and $R = \mathbf{0}_{n \times p}$

- The gain matrix $K_1 \in \mathbb{R}^{n \times p}$ is **not unique** due to the multivariable nature of the problem. Thus, **specific properties** may be investigated, such as find K_1 which gives directional or minimum variance residuals

Theory of Unknown Input Observers (UIO)

Unknown Input Observer (UIO)

- Check the rank condition $\text{rank}(CB_d) = \text{rank}(B_d)$
- Compute $R = B_d[(CB_d)^\top CB_d]^{-1}(CB_d)^\top \qquad T = I_n - RC \qquad A_1 = TA$
- If (C, A_1) observable, compute K_1 by pole placement to assign the eigenvalues of $A_1 - K_1 C = F$
- Compute $F = A_1 - K_1 C \qquad K_2 = FR \qquad K = K_1 + K_2$

UIO residual generator

Consider the following LTI system with additive disturbances and faults

$$\begin{cases} \underset{n\times 1}{\boldsymbol{x}(t+1)} = \underset{n\times n}{A}\cdot\boldsymbol{x}(t) + \underset{n\times m_u\ m_u\times 1}{B\cdot\boldsymbol{u}(t)} + \underset{n\times m_d\ m_d\times 1}{B_d\cdot\boldsymbol{d}(t)} + \underset{n\times m_f\ m_f\times 1}{B_f\cdot\boldsymbol{f}(t)} \\ \underset{p\times 1}{\boldsymbol{y}(t)} = \underset{p\times n}{C}\cdot\boldsymbol{x}(t) + \underset{p\times m_f}{D_f}\cdot\boldsymbol{f}(t) \end{cases}$$

A residual can be computed by the UIO as follows **(UIO-based residual generator)**

$$\begin{cases} \underset{n\times 1}{\boldsymbol{m}(t+1)} = \underset{n\times n\ \ n\times 1}{F\cdot\boldsymbol{m}(t)} + \underset{n\times p\ \ p\times 1}{K\cdot\boldsymbol{y}(t)} + \underset{n\times n\ \ n\times m_u\ \ m_u\times 1}{TB\cdot\boldsymbol{u}(t)} \\ \underset{n\times 1}{\hat{\boldsymbol{x}}(t)} = \underset{n\times 1}{\boldsymbol{m}(t)} + \underset{n\times p\ \ p\times 1}{R\cdot\boldsymbol{y}(t)} \\ \boldsymbol{r}(t) = \boldsymbol{y}(t) - C\cdot\hat{\boldsymbol{x}}(t) = (I-CR)\boldsymbol{y}(t) - C\boldsymbol{m}(t) \end{cases}$$

UIO residual generator

The UIO-based residual generator

$$\begin{cases} \underset{n\times 1}{\boldsymbol{m}(t+1)} = \underset{n\times n\ \ n\times 1}{F\cdot\boldsymbol{m}(t)} + \underset{n\times p\ \ p\times 1}{K\cdot\boldsymbol{y}(t)} + \underset{n\times n\ \ n\times m_u\ \ m_u\times 1}{TB\cdot\boldsymbol{u}(t)} \\ \boldsymbol{r}(t) = \boldsymbol{y}(t) - C\cdot\hat{\boldsymbol{x}}(t) = (I-CR)\cdot\boldsymbol{y}(t) - C\cdot\boldsymbol{m}(t) \end{cases}$$

resembles the more general observer-based one

$$\begin{cases} \underset{n_m\times 1}{\boldsymbol{m}(t+1)} = \underset{n_m\times n_m}{F\cdot\boldsymbol{m}(t)} + \underset{n_m\times p\ \ p\times 1}{K\cdot\boldsymbol{y}(t)} + \underset{n_m\times m_u}{J\cdot\boldsymbol{u}(t)} \\ \underset{q\times 1}{\boldsymbol{r}(t)} = \underset{q\times n_m}{L_1\cdot\boldsymbol{m}(t)} + \underset{q\times p}{L_2\cdot\boldsymbol{y}(t)} + \underset{q\times m_u}{L_3\cdot\boldsymbol{u}(t)} \end{cases}$$

- $J = TB$
- $L_1 = -C$
- $L_2 = (I - CR)$
- $L_3 = \mathbf{0}$
- $F = A_1 - K_1 C$

UIO residual generator

The **computational form** of residuals $\boldsymbol{r}(t)$ can be expressed by the **transfer function matrix** $\boldsymbol{Q}(z)$ as

$$\underset{q\times(m_u+p)}{\boldsymbol{r}(t) = \boldsymbol{Q}(z)} \underset{(m_u+p)\times 1}{\begin{bmatrix}\boldsymbol{u}(t)\\ \boldsymbol{y}(t)\end{bmatrix}} = [Q_u(z) \quad Q_y(z)]\begin{bmatrix}\boldsymbol{u}(t)\\ \boldsymbol{y}(t)\end{bmatrix}$$

General observer case (computational form)

$$\boldsymbol{r}(t) = [L_1(zI-F)^{-1}J + L_3]\cdot\boldsymbol{u}(t) + [L_1(zI-F)^{-1}K + L_2]\cdot\boldsymbol{y}(t)$$

UIO case (computational form)

$$\boldsymbol{r}(t) = [-C(zI-F)^{-1}TB]\cdot\boldsymbol{u}(t) + [-C(zI-F)^{-1}K + (I-CR)]\cdot\boldsymbol{y}(t)$$

UIO residual generator

Applying the UIO-based residual generator to the system we have

$$\boldsymbol{\varepsilon_x}(t+1) \equiv \boldsymbol{x}(t+1) - \hat{\boldsymbol{x}}(t+1)$$
$$= A\boldsymbol{x}(t) + B\boldsymbol{u}(t) + B_d\boldsymbol{d}(t) + B_f\boldsymbol{f}(t) - \boldsymbol{m}(t+1) - R\boldsymbol{y}(t+1)$$
$$= A\boldsymbol{x}(t) + B\boldsymbol{u}(t) + B_d\boldsymbol{d}(t) + B_f\boldsymbol{f}(t) - F\boldsymbol{m}(t) - TB\boldsymbol{u}(t) - K\boldsymbol{y}(t) - R\boldsymbol{y}(t+1)$$

$$R\boldsymbol{y}(t+1) = RC\boldsymbol{x}(t+1) + RD_f\boldsymbol{f}(t)$$
$$= RCA\boldsymbol{x}(t) + RCB\boldsymbol{u}(t) + RCB_d\boldsymbol{d}(t) + RCB_f\boldsymbol{f}(t) + RD_f\boldsymbol{f}(t+1)$$

$$F\boldsymbol{m}(t) = F[\hat{\boldsymbol{x}}(t) - R\boldsymbol{y}(t)] = F\hat{\boldsymbol{x}}(t) - FR\boldsymbol{y}(t) = F\hat{\boldsymbol{x}}(t) - K_2\boldsymbol{y}(t)$$

UIO residual generator

$$\boldsymbol{\varepsilon_x}(t+1) = A\boldsymbol{x}(t) + B\boldsymbol{u}(t) + B_d\boldsymbol{d}(t) + B_f\boldsymbol{f}(t) - TB\boldsymbol{u}(t) - K\boldsymbol{y}(t) +$$
$$-F\hat{\boldsymbol{x}}(t) + K_2\boldsymbol{y}(t) +$$
$$-RCA\boldsymbol{x}(t) - RCB\boldsymbol{u}(t) - RCB_d\boldsymbol{d}(t) - RCB_f\boldsymbol{f}(t) - RD_f\boldsymbol{f}(t+1)$$

A_1 $\qquad$ $(I - RC)B_d = 0$ by UIO design

$$= (I - RC)A\boldsymbol{x}(t) + (I - RC)B\boldsymbol{u}(t) + (I - RC)B_d\boldsymbol{d}(t) + (I - RC)B_f\boldsymbol{f}(t) - TB\boldsymbol{u}(t)$$
$$- K\boldsymbol{y}(t) - F\hat{\boldsymbol{x}}(t) + K_2\boldsymbol{y}(t) - RD_f\boldsymbol{f}(t+1)$$

$$= A_1\boldsymbol{x}(t) + TB_f\boldsymbol{f}(t) - K\boldsymbol{y}(t) - F\hat{\boldsymbol{x}}(t) + K_2\boldsymbol{y}(t) - RD_f\boldsymbol{f}(t+1)$$

$$= A_1\boldsymbol{x}(t) + TB_f\boldsymbol{f}(t) - (K_1 + K_2)\boldsymbol{y}(t) - F\hat{\boldsymbol{x}}(t) + K_2\boldsymbol{y}(t) - RD_f\boldsymbol{f}(t+1)$$

UIO residual generator

$$\boldsymbol{\varepsilon_x}(t+1) = A_1\boldsymbol{x}(t) + TB_f\boldsymbol{f}(t) - K_1\boldsymbol{y}(t) - F\hat{\boldsymbol{x}}(t) - RD_f\boldsymbol{f}(t+1)$$
$$= A_1\boldsymbol{x}(t) + TB_f\boldsymbol{f}(t) - K_1[C\boldsymbol{x}(t) + D_f\boldsymbol{f}(t)] - F\hat{\boldsymbol{x}}(t) - RD_f\boldsymbol{f}(t+1)$$
F
$$= [A_1 - K_1C]\boldsymbol{x}(t) + TB_f\boldsymbol{f}(t) - F\hat{\boldsymbol{x}}(t) - K_1D_f\boldsymbol{f}(t) - RD_f\boldsymbol{f}(t+1)$$
$$= F\boldsymbol{\varepsilon_x}(t) + TB_f\boldsymbol{f}(t) - K_1D_f\boldsymbol{f}(t) - RD_f\boldsymbol{f}(t+1)$$

The residual $\boldsymbol{r}(t)$ can be expressed as function of $\boldsymbol{\varepsilon_x}(t)$ as

$$\boldsymbol{r}(t) = \boldsymbol{y}(t) - \hat{\boldsymbol{y}}(t) = \boldsymbol{y}(t) - C\hat{\boldsymbol{x}}(t) = C\boldsymbol{x}(t) + D_f\boldsymbol{f}(t) - C\hat{\boldsymbol{x}}(t) = C\boldsymbol{\varepsilon_x}(t) + D_f\boldsymbol{f}(t)$$

UIO residual generator

When faults are present, the residual $\boldsymbol{r}(t)$ and the state estimation error $\boldsymbol{\varepsilon_x}(t)$ are related by

$$\begin{cases} \underset{n\times 1}{\boldsymbol{\varepsilon_x}(t+1)} = \underset{n\times n}{F}\cdot\underset{n\times 1}{\boldsymbol{\varepsilon_x}(t)} + \underset{n\times n}{TB_f}\cdot\underset{n\times m_f}{\boldsymbol{f}(t)} - \underset{n\times p}{K_1}\underset{p\times m_f}{D_f}\cdot\underset{n\times p}{\boldsymbol{f}(t)} - \underset{n\times p}{RD_f}\cdot\underset{p\times m_f}{\boldsymbol{f}(t+1)} \\ \underset{p\times 1}{\boldsymbol{r}(t)} = \underset{p\times n}{C}\cdot\underset{n\times 1}{\boldsymbol{\varepsilon_x}(t)} + \underset{p\times m_f}{D_f}\cdot\underset{m_f\times 1}{\boldsymbol{f}(t)} \end{cases}$$

This system is however not so useful to analyze due to the term $RD_f\boldsymbol{f}(t+1)$. We thus define an alternative system (but equivalent as regards the output $\boldsymbol{r}(t)$), by defining

$$\boldsymbol{e}(t+1) = \boldsymbol{\varepsilon_x}(t+1) + RD_f\boldsymbol{f}(t+1)$$

UIO residual generator

Substituing $\boldsymbol{e}(t)$ in place of $\boldsymbol{\varepsilon_x}(t)$ we get

$$\begin{cases} \boldsymbol{e}(t+1) = F \cdot \boldsymbol{e}(t) + TB_f \cdot \boldsymbol{f}(t) - K_1 D_f \cdot \boldsymbol{f}(t) - FRD_f \boldsymbol{f}(t) \\ \boldsymbol{r}(t) = C \cdot \boldsymbol{e}(t) + (I - CR) D_f \cdot \boldsymbol{f}(t) \end{cases}$$

Which resembles the more general observer-based one

$$\begin{cases} \underset{n_m \times 1}{\boldsymbol{\varepsilon_m}(t+1)} = \underset{n_m \times n_m}{F} \cdot \underset{n_m \times n \; n \times m_f}{\boldsymbol{\varepsilon_m}\,(t)} - \underset{n_m \times p \; p \times m_f}{TB_f} \cdot \boldsymbol{f}(t) + \underset{n_m \times p \; p \times m_f}{KD_f} \cdot \boldsymbol{f}(t) \\ \underset{q \times 1}{\boldsymbol{r}(t)} = \underset{q \times n_m}{L_1} \cdot \boldsymbol{\varepsilon_m}(t) + \underset{q \times p \quad p \times m_f}{L_2 D_f} \cdot \underset{m_f \times 1}{\boldsymbol{f}(t)} \end{cases}$$

- $L_1 = -C$
- $L_2 = (I - CR)$

UIO residual generator

The **internal form** of residuals $\boldsymbol{r}(t)$ can be expressed as

General observer case (internal form)

$$\boldsymbol{r}(t) = \left[L_1 (zI - F)^{-1} \cdot \left(-TB_f + KD_f\right) + L_2 D_f\right] \cdot \boldsymbol{f}(t)$$

UIO case (internal form)

$$\begin{aligned} \boldsymbol{r}(t) &= \left[C(zI - F)^{-1} \cdot \left(+TB_f - KD_f\right) + (I - CR) D_f\right] \cdot \boldsymbol{f}(t) \\ &= \left[-C(zI - F)^{-1} \cdot \left(-TB_f + KD_f\right) + (I - CR) D_f\right] \cdot \boldsymbol{f}(t) \end{aligned}$$

- $L_1 = -C$
- $L_2 = (I - CR)$

Example: robust UIO residual generation

Consider the following MIMO system with $m_u = 2, p = 3, m_f = 1$, sampled at $T_s = 1$ s.

Assume there is $m_d = 1$ disturbances acting on states and outputs

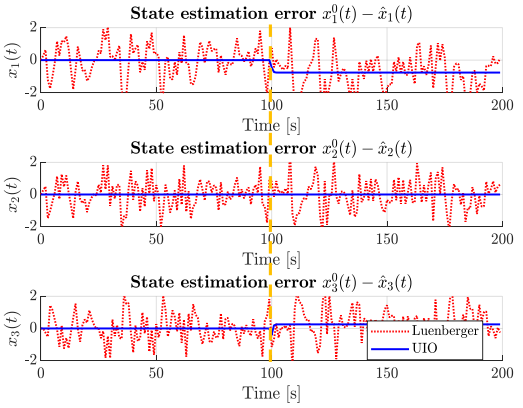
$$\underset{n \times n \; 4 \times 4}{A} = \begin{bmatrix} 0.5 & -0.7 & 0.7 & 0 \\ 0 & 0.8 & 0.06 & 0 \\ -1 & 0 & 0 & 0.1 \\ 0 & 0 & -0.1 & 0.4 \end{bmatrix} \quad \underset{n \times m_u \; 4 \times 2}{B} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \quad \underset{p \times n \; 3 \times 4}{C} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \underset{p \times m_u \; 3 \times 2}{D} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

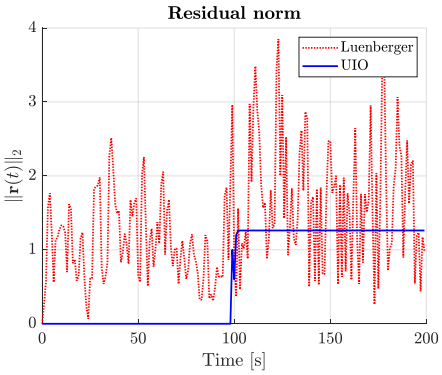
$$\underset{n \times m_f \; 4 \times 1}{B_f} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \underset{p \times m_f \; 3 \times 1}{D_f} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \quad \underset{n \times m_d \; 4 \times 1}{B_d} = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} \quad \underset{p \times m_d \; 3 \times 1}{D_d} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

- Observers poles: 0.0729, 0.197, 0.0053, 0.27
- $d(t) \sim \mathrm{WN}(0,1)$

Collect $N = 200$ data. The inputs are white noises with zero mean and variance 1. A step-like fault of amplitude 1 acts on the system at time $t = 100$ s.

Example: robust UIO residual generation





UNIVERSITÀ DEGLI STUDI DI BERGAMO | Dipartimento di Ingegneria Gestionale, dell'Informazione e della Produzione

UNIVERSITÀ DEGLI STUDI DI BERGAMO | Dipartimento di Ingegneria Gestionale, dell'Informazione e della Produzione

ADAPTIVE LEARNING, ESTIMATION AND SUPERVISION OF DYNAMICAL SYSTEMS (ALES)

Lecture 15: Fault isolation

Master Degree in COMPUTER ENGINEERING

Data Science and Data Engineering Curriculum

SPEAKER
Prof. Mirko Mazzoleni

PLACE
University of Bergamo

Syllabus

1. **Recursive and adaptive identification**
   - 1.1 Recursive ARX estimation (RLS)
   - 1.2 Least Mean Squares (LMS)
   - 1.3 Instrumental Variables (IV)
2. **Closed-loop identification**
3. **Subspace and MIMO identification**
   - 3.1 Singular Value Decomposition
   - 3.2 Impulse data: Ho-Kalman, Kung algorithms
   - 3.3 Generic I/O data: the MOESP algorithm
4. **Supervision of dynamical systems**
   - 4.1 Introduction to fault diagnosis
   - 4.2 Model-based fault diagnosis
   - 4.3 Parity space approaches
   - 4.4 Observer-based approaches
   - 4.5 Signal-based fault diagnosis
   - 4.6 Knowledge-based fault diagnosis

CASE STUDIES

- Virtual Reference Feedback Tuning
- Nuclear particles classification
- Leak detection in an industrial valve
- Bearing fault identification

Outline

1. Fault isolation concepts
2. Single sensor parity relation (SSPR)
3. Observer-based isolation schemes

Outline

1. **Fault isolation concepts**
2. Single sensor parity relation (SSPR)
3. Observer-based isolation schemes

Fault isolation concepts

A **fault isolation** procedure has the aim of **distinguish** (isolate) a particular fault from other faults

Usually (but not necessary), a **set of residuals** is required for fault isolation

Two main approaches exist for designing residuals with fault isolation properties:

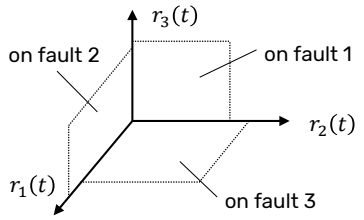
1. **Structured** residual set
2. **Directional** residual set

Structured residual set

Idea: Each residual is designed to be **sensitive** to a **subset of faults**, whilst insensitive to the remaining faults

When a particular fault occurs, some residuals respond and some not. Thus, the **boolean pattern** of response set **(fault signature** or **code)** characterizes the fault

Fault isolation amounts to **comparing** the **actual** obtained fault code with a **predefined** set of fault codes



Structured residual set

DEDICATED RESIDUAL SET

A **dedicated** residual set is a structured residual set where the i-th residual $r_i(t)$ is sensitive only to the i-th fault $f_i(t)$

$$r_i(t) = J(f_i(t)), \qquad i \in \{1,2,\dots,m_f\}$$

$J(\ )$ is a generic functional relationship

Given the i-th threshold η_i, a fault isolation **decision logic** is

If $r_i(t) > \eta_i$ **then** $f_i(t) \neq 0 \quad i \in \{1,2,\dots,m_f\}$

This isolable residual structure is very **simple**, and all faults can be detected **simultaneously**. However, it is difficult to design in practice

Structured residual set

GENERALIZED RESIDUAL SET

A **generalized** residual set is a structured residual set where the i-th residual $r_i(t)$ is sensitive to all but the i-th fault $f_i(t)$

$$\begin{cases} r_1(t) = J\left(f_2(t), \dots, f_{m_f}(t)\right) & r_1(t) \text{ does not depend on } f_1(t) \\ \vdots & \\ r_i(t) = J\left(f_1(t), \dots, f_{i-1}(t), f_{i+1}(t), \dots, f_{m_f}(t)\right) & r_i(t) \text{ does not depend on } f_i(t) \\ \vdots & \\ r_{m_f}(t) = J\left(f_1(t), \dots, f_{m_f-1}(t)\right) & r_{m_f}(t) \text{ does not depend on } f_{m_f}(t) \end{cases}$$

Structured residual set

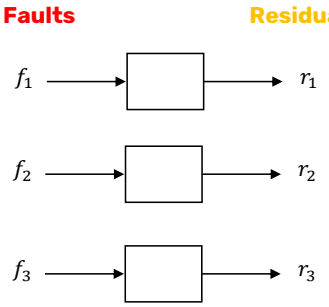
GENERALIZED RESIDUAL SET

Given the i-th threshold η_i, a fault isolation **decision logic** is

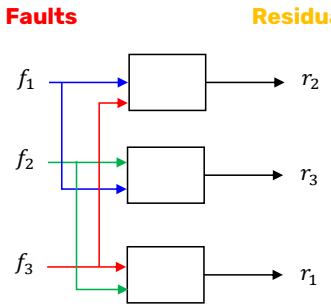
$$\textbf{If} \begin{cases} r_i(t) \le \eta_i \\ r_j(t) > \eta_j \quad \forall j \in \{1, \dots, i-1, i+1, \dots, m_f\} \end{cases} \quad \textbf{then} \quad f_i(t) \neq 0$$

Structured residual set

DEDICATED RESIDUAL SET



GENERALIZED RESIDUAL SET



Structured residual set

More generally, subdivide the residual vector $\boldsymbol{r}(t)$ into **residual sets** $\boldsymbol{r}^{[i]}(t)$, so that

$$\underset{q \times 1}{\boldsymbol{r}(t)} = \begin{bmatrix} \boldsymbol{r}^{[1]}(t) \\ \boldsymbol{r}^{[2]}(t) \\ \vdots \\ \boldsymbol{r}^{[n_b]}(t) \end{bmatrix} \begin{matrix} q_1 \times 1 \\ q_2 \times 1 \\ \\ q_{n_b} \times 1 \end{matrix}$$

The residual set $\boldsymbol{r}^{[i]}$ can be **evaluated** by an evaluation function $J(\ \ )$ and then **compared** with a scalar threshold η_i, obtaining the **Boolean decision** vector $\boldsymbol{\iota}(t) \in \mathbb{R}^{n_b \times 1}$

$$\underset{n_b \times 1}{\boldsymbol{\iota}(t)} := \begin{bmatrix} i_1(t) \\ i_2(t) \\ \vdots \\ \iota_{n_b}(t) \end{bmatrix}$$

Structured residual set

A structured residual set relies on the definition of a **structure matrix** S_{res} that specifies the **residual set** sensitivities and insensitivities to faults

$$S_{\text{res}} = \underset{n_b \times m_f}{\left[\;\right]}$$

i -th row: **specification** associated to residual set $\boldsymbol{r}^{[i]}(t)$

j -th column: **signature** or **code** associated to fault $f_j(t)$

The isolation is perfomed by comparing $\boldsymbol{\iota}(t)$ with each **fault signature**

Structured residual set

For instance, consider a problem with $m_f = 3$ faults. Examples of structure matrices are:

$S_{\text{res}} = [1\ 1\ 1]$ **Complete fault detectability.** Allows the detection (but no isolation) of any fault or an arbitrary number of simultaneous faults ($n_b = 1$)

$S_{\text{res}} = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}$ **Weak fault isolability** *(generalized residual set).* Allows to isolate any individual fault occurring one at a time ($n_b = m_f$). Can cope with partial firings of the residuals, without producing false alarms

$S_{\text{res}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ **Strong fault isolability** *(dedicated residual set).* Allows to isolate an arbitrary number of simultaneous faults ($n_b = m_f$)

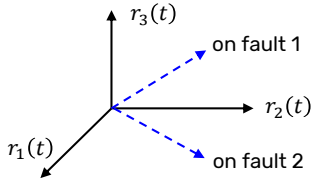
Directional residual set

Idea: Each residual vector is designed to **lie** in a fixed and fault-specific **direction** (or subspace) in the **residual space**, in response to a particular fault, so that

$$\boldsymbol{r}\big(t|f_i(t)\big) = \alpha_i(t)\boldsymbol{l}_i, \qquad i \in \{1,2,\dots,m_f\}$$

where the constant vector $\boldsymbol{l}_i$ is the **signature direction** of the i-th fault in the residual space and $\alpha_i(t) \coloneqq \gamma_i(z)f_j(t)$ is a scalar that depends on the fault size and dynamics

Fault isolation amounts determining which of the **known fault signature directions** the generated residual vector **lies the closest to**



Example: static hydraulic system

Consider the example shown in Lecture 10 slide 23, only with actuator and sensors faults

$$y_1(t) = u_a(t) + f_a(t) + f_{y_1}(t) \qquad y_2(t) = u_a(t) + f_a(t) + f_{y_2}(t)$$

In this static system example, the fault transfer matrix $\boldsymbol{G}_f(z)$ is just a gain matrix that shows the residuals sensitivities to faults

$$\boldsymbol{G}_f(z) = \begin{matrix} f_a \\ \downarrow \end{matrix} \begin{matrix} f_{y_1} \\ \downarrow \end{matrix} \begin{matrix} f_{y_2} \\ \downarrow \end{matrix} \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

- In response to a fault f_a, the residual vector is confined to the $+45°$ **line**
- In response to a fault f_{y_1}, the residual vector is confined to the r_1 **axis**
- In response to a fault f_{y_2}, the residual vector is confined to the r_2 **axis**

Sensor faults isolation

If only **sensor faults** are of interest, the system output is given by (in absence of disturbances, noise and modeling errors)

$$\boldsymbol{y}(t) = \boldsymbol{G}_u^0(z)\boldsymbol{u}(t) + \boldsymbol{f}_y$$

Assume $p = m_{f_y}$ for simplicity

If one wants to design a residual signal which is **sensitive to the group of faults** in $\boldsymbol{f}_y^{[1]}(t)$ and insensitive to faults in group $\boldsymbol{f}_y^{[2]}(t)$, we can decompose the output equation as

$$\begin{bmatrix}\boldsymbol{y}^{[1]}(t)\\ \boldsymbol{y}^{[2]}(t)\end{bmatrix} = \boldsymbol{G}_u^0(z)\boldsymbol{u}(t) + \begin{bmatrix}\boldsymbol{f}_y^{[1]}(t)\\ \boldsymbol{f}_y^{[2]}(t)\end{bmatrix}$$

Sensor faults isolation

The residual generator for $\boldsymbol{f}_y^{[1]}$ then takes the form (similarly for $\boldsymbol{f}_y^{[2]}$)

$$\boldsymbol{r}^{[1]}(t) = \boldsymbol{Q}_u^{[1]}(z)\boldsymbol{u}(t) + \boldsymbol{Q}_y^{[1]}(z)\boldsymbol{y}^{[1]}(t) = \boldsymbol{Q}_u^{[1]}(z)\boldsymbol{u}(t) + \boldsymbol{Q}_y^{[1]}(z)\left[\boldsymbol{G}_u^0(z)\boldsymbol{u}(t) + \boldsymbol{f}_y^{[1]}\right] =$$

$$= \left[\boldsymbol{Q}_u^{[1]}(z) + \boldsymbol{Q}_y^{[1]}(z)\boldsymbol{G}_u^0(z)\right]\boldsymbol{u}(t) + \boldsymbol{Q}_y^{[1]}(z)\boldsymbol{f}_y^{[1]}$$

The residual is driven by **all inputs** and only a **subset of outputs**

Thus, if

$$\begin{cases}\boldsymbol{Q}_u^{[1]}(z) = -\boldsymbol{Q}_y^{[1]}(z)\boldsymbol{G}_u^0(z)\\ \boldsymbol{Q}_y^{[1]}(z) \neq \boldsymbol{0}\end{cases}$$

These are the standard requirements for a residual generator, see Lecture 10 Slide 35

The residual $\boldsymbol{r}^{[1]}(t)$ will be **only sensitive** to faults in the group $\boldsymbol{f}_y^{[1]}$

Actuator faults isolation

When **actuator faults** occur in the system (without disturbances and noises, and assuming no measured inputs) we have

$$\boldsymbol{y}(t) = \boldsymbol{G}_a^0(z)\boldsymbol{u}_a^0(t) = \boldsymbol{G}_a^0(z)[\boldsymbol{u}_a(t) + \boldsymbol{f}_a(t)]$$

Assume $m_{u_a} = m_{f_a}$ for simplicity

If one wants to design a residual signal which is **sensitive to the group of faults** in $\boldsymbol{f}_a^{[1]}(t)$ and insensitive to faults in group $\boldsymbol{f}_a^{[2]}(t)$, we can decompose the output equation as

$$\boldsymbol{y}(t) = \begin{bmatrix}\boldsymbol{G}_a^{0\,[1]}(z) & \boldsymbol{G}_a^{0\,[2]}(z)\end{bmatrix}\begin{bmatrix}\boldsymbol{u}_a{}^{[1]}(t) + \boldsymbol{f}_a^{[1]}(t)\\ \boldsymbol{u}_a{}^{[2]}(t) + \boldsymbol{f}_a^{[2]}(t)\end{bmatrix}$$

$$= \boldsymbol{G}_a^{0\,[1]}(z)\left[\boldsymbol{u}_a{}^{[1]}(t) + \boldsymbol{f}_a^{[1]}(t)\right] + \boldsymbol{G}_a^{0\,[2]}(z)\left[\boldsymbol{u}_a{}^{[2]}(t) + \boldsymbol{f}_a^{[2]}(t)\right]$$

Actuator faults isolation

The residual generator for $\boldsymbol{f}_a^{[1]}$ then takes the form (similarly for $\boldsymbol{f}_y^{[2]}$)

$$\boldsymbol{r}^{[1]}(t) = \boldsymbol{Q}_u^{[1]}(z)\boldsymbol{u}_a^{[1]}(t) + \boldsymbol{Q}_y^{[1]}(z)\boldsymbol{y}(t)$$

The residual is driven by **all outputs** and only a **subset of inputs**

$$= \boldsymbol{Q}_u^{[1]}(z)\boldsymbol{u}_a^{[1]}(t) + \boldsymbol{Q}_y^{[1]}(z)\left[\boldsymbol{G}_a^{0\,[1]}(z)\left(\boldsymbol{u}_a^{[1]}(t) + \boldsymbol{f}_a^{[1]}(t)\right) + \boldsymbol{G}_a^{0\,[2]}(z)\left(\boldsymbol{u}_a^{[2]}(t) + \boldsymbol{f}_a^{[2]}(t)\right)\right]$$

$$= \left[\boldsymbol{Q}_u^{[1]}(z) + \boldsymbol{Q}_y^{[1]}(z)\boldsymbol{G}_a^{0\,[1]}(z) + \boldsymbol{Q}_y^{[1]}(z)\boldsymbol{G}_a^{0\,[2]}\right]\boldsymbol{u}_a^{[1]}(t) + \boldsymbol{Q}_y^{[1]}(z)\boldsymbol{G}_a^{0\,[1]}(z)\boldsymbol{f}_a^{[1]}(t)$$

$$+ \boldsymbol{Q}_y^{[1]}(z)\boldsymbol{G}_a^{0\,[2]}(z)\left(\boldsymbol{u}_a^{[2]}(t) + \boldsymbol{f}_a^{[2]}(t)\right)$$

Actuator faults isolation

Thus,

$$\begin{cases} \boldsymbol{Q}_u^{[1]}(z) = -\boldsymbol{Q}_y^{[1]}(z)\boldsymbol{G}_a^{0\,[1]}(z) \\ \boldsymbol{Q}_y^{[1]}(z)\boldsymbol{G}_a^{0\,[2]}(z) = \boldsymbol{0} \\ \boldsymbol{Q}_y^{[1]}(z)\boldsymbol{G}_a^{0\,[1]}(z) \neq \boldsymbol{0} \end{cases}$$

There is an extra constraint $\boldsymbol{Q}_y^{[1]}(z)\boldsymbol{G}_a^{0\,[2]}(z) = \boldsymbol{0}$

A stable and implementable transfer matrix $\boldsymbol{Q}_y^{[1]}$ which satisfies the additional constraint $\boldsymbol{Q}_y^{[1]}(z)\boldsymbol{G}_a^{0\,[2]}(z) = \boldsymbol{0}$ does not always exist

Hence, **actuator fault isolation is not always possible**

Outline

1. Fault isolation concepts

2. Single sensor parity relation (SSPR)

3. Observer-based isolation schemes

Single sensor parity relation

The design for isolating sensor faults in the parity space framework is very straightforward

Let $\boldsymbol{c}_i^\top \in \mathbb{R}^{1\times n}$ the i-th row of the output matrix $C \in \mathbb{R}^{p\times n}$, and $y_i(t) \in \mathbb{R}$ the i-th component of the output vector $\boldsymbol{y}(t) \in \mathbb{R}^{p\times 1}$

If $\boldsymbol{c}_i^\top$ and $y_i(t)$ are used in place of C and $\boldsymbol{y}(t)$, the parity relation will **contain only** the i-th **sensor's output**, together will all inputs

Thus, a set of parity relation can be constructed, each sensitive to only one sensor fault

Outline

1. Fault isolation concepts

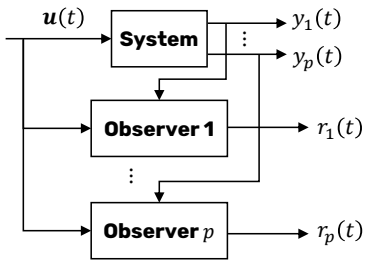
2. Single sensor parity relation (SSPR)

3. Observer-based isolation schemes

Observer-based isolation schemes

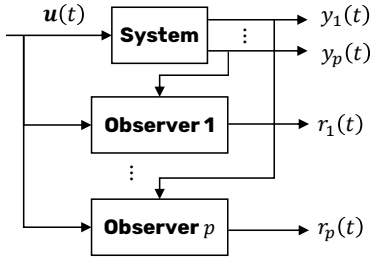
DEDICATED OBSERVER SCHEME

Bank of observers, each excited by a single different output



GENERALIZED OBSERVER SCHEME

Bank of observers, each excited by all but one outputs



Generalized observer for sensor faults using UIO

Consider the following system without noises and with **only sensor faults**

$$\begin{cases} \underset{n\times 1}{\boldsymbol{x}(t+1)} = \underset{n\times n}{A}\cdot\underset{n\times 1}{\boldsymbol{x}(t)} + \underset{n\times m_u}{B}\cdot\underset{m_u\times 1}{\boldsymbol{u}(t)} + \underset{n\times m_d}{B_d}\cdot\underset{m_d\times 1}{\boldsymbol{d}(t)} \\ \underset{(p-1)\times 1}{\boldsymbol{y}^{[j]}(t)} = \underset{(p-1)\times n}{C^{[j]}}\cdot\boldsymbol{x}(t) + \underset{(p-1)\times 1}{\boldsymbol{f}_y^{[j]}(t)} \\ \underset{1\times 1}{y_j(t)} = \underset{1\times n}{\boldsymbol{c}_j^{\mathsf{T}}}\cdot\boldsymbol{x}(t) + f_{y,j}(t) \end{cases} \quad \textbf{for } j = 1,\dots,p$$

- $\boldsymbol{y}^{[j]} \in \mathbb{R}^{(p-1)\times 1}$ is obtained from the output vector $\boldsymbol{y}(t)$ with the j-th row removed
- $y_j \in \mathbb{R}^{1\times 1}$ the j-th component of $\boldsymbol{y}(t)$
- $C^{[j]} \in \mathbb{R}^{(p-1)\times n}$ is obtained from the C matrix with the j-th row $\boldsymbol{c}_j^{\mathsf{T}}$ removed

For ease of notation, we assume that $\boldsymbol{f}_y(t)$ has p components. However, all the discussion holds by considering the component $S_y\boldsymbol{f}_y(t)$ if $m_{f_y} < p$

Generalized observer for sensor faults using UIO

Based on the previous system description, we can design p **UIO residual generators** as

$$\begin{cases} \underset{n\times 1}{\boldsymbol{m}^{[j]}(t+1)} = \underset{n\times n}{F^{[j]}}\cdot\underset{n\times 1}{\boldsymbol{m}^{[j]}(t)} + \underset{n\times(p-1)}{K^{[j]}}\cdot\overset{(p-1)\times 1}{\boldsymbol{y}^{[j]}(t)} + \underset{n\times n}{T^{[j]}}\underset{n\times m_u}{B}\cdot\underset{m_u\times 1}{\boldsymbol{u}(t)} \\ \underset{(p-1)\times 1}{\boldsymbol{r}^{[j]}(t)} = \underset{n\times(p-1)}{\left(I - C^{[j]}R^{[j]}\right)}\cdot\underset{(p-1)\times n}{\boldsymbol{y}^{[j]}(t)} - \underset{(p-1)\times n}{C^{[j]}}\cdot\boldsymbol{m}^{[j]}(t) \end{cases} \quad \textbf{for } j = 1,\dots,p$$

The matrices must satisfy the following equations

$$\left.\begin{aligned} R^{[j]}C^{[j]}B_d &= B_d \\ T^{[j]} &= I - R^{[j]}C^{[j]} \\ F^{[j]} &= T^{[j]}A - K_1^{[j]}C^{[j]} \quad \text{to be stabilized} \\ K_2^{[j]} &= F^{[j]}H^{[j]} \\ K^{[j]} &= K_1^{[j]} + K_2^{[j]} \end{aligned}\right\} \quad \textbf{for } j = 1,\dots,p$$

Generalized observer for sensor faults using UIO

Each residual generator of this type is **driven by all inputs and all but one output**

When all **actuators are fault-free** and a **fault occurs in the** j-th **sensor**, the residuals will satisfy the following isolation logic:

$$\begin{cases} \left\|\boldsymbol{r}^{[j]}(t)\right\| \le \eta_j^{\text{sensor}} \\ \left\|\boldsymbol{r}^{[k]}(t)\right\| > \eta_k^{\text{sensor}} \quad \text{for} \quad k = 1,\dots,j-1,j+1,\dots,p \end{cases}$$

Generalized observer for actuator faults using UIO

Consider the following system without noises and with **only actuator faults** and no measured inputs

$$\begin{cases} \boldsymbol{x}(t+1) = A\boldsymbol{x}(t) + \overbrace{B^{[i]}\left(\boldsymbol{u}_a^{[i]}(t) + \boldsymbol{f}_a^{[i]}(t)\right)}^{B^{[i]}\boldsymbol{u}_a^{0\,[i]}(t)} + \boldsymbol{b}_i\left(u_{a,i}(t) + f_{a,i}(t)\right) + B_d\boldsymbol{d}(t) \\ \qquad\qquad = A\boldsymbol{x}(t) + B^{[i]}\boldsymbol{u}_a^{[i]}(t) + B^{[i]}\boldsymbol{f}_a^{[i]}(t) + B_d^{[i]}\boldsymbol{d}^{[i]}(t) \\ \boldsymbol{y}(t) = C\boldsymbol{x}(t) + \boldsymbol{f}_y(t) \end{cases} \quad \textbf{for } i = 1, \dots, m_u$$

The i-th UIO is insensitive to the i-th actuator fault

- $\boldsymbol{u}_a^{[i]} \in \mathbb{R}^{(m_u-1)\times 1}$ is obtained from the input vector $\boldsymbol{u}(t)$ with the i-th row removed
- $u_{a,i} \in \mathbb{R}^{1\times 1}$ the i-th component of $\boldsymbol{u}_a(t)$
- $B^{[i]} \in \mathbb{R}^{n\times(m_u-1)}$ is obtained from the B matrix with the i-th column $\boldsymbol{b}_i$ removed
- $B_d^{[i]} \coloneqq [B_d \quad b_i] \in \mathbb{R}^{n\times(m_d+1)}$, $\quad \boldsymbol{d}^{[i]}(t) \coloneqq \begin{bmatrix} \boldsymbol{d}(t) \\ u_{a,i}(t) + f_{a,i}(t) \end{bmatrix}$

Generalized observer for actuator faults using UIO

Based on the previous system description, we can design m_u **UIO residual generators** as

$$\begin{cases} \underset{n\times 1}{\boldsymbol{m}^{[i]}}(t+1) = \underset{n\times n}{F^{[i]}} \cdot \boldsymbol{m}^{[i]}(t) + \underset{n\times p}{K^{[i]}} \cdot \boldsymbol{y}(t) + \underset{n\times n}{T^{[i]}}\underset{n\times(m_u-1)}{B^{[i]}} \cdot \underset{(m_u-1)\times 1}{\boldsymbol{u}_a^{[i]}}(t) \\ \underset{p\times 1}{\boldsymbol{r}^{[j]}}(t) = \left(I - \underset{n\times p}{CR^{[j]}}\right) \cdot \boldsymbol{y}(t) - \underset{p\times n}{C} \cdot \boldsymbol{m}^{[j]}(t) \end{cases} \quad \textbf{for } i = 1, \dots, m_u$$

The matrices must satisfy the following equations

$$\left.\begin{aligned} R^{[i]}CB_d^{[i]} &= B_d^{[i]} \\ T^{[i]} &= I - R^{[i]}C \\ F^{[i]} &= T^{[i]}A - K_1^{[i]}C \quad \text{to be stabilized} \\ K_2^{[i]} &= F^{[i]}H^{[i]} \\ K^{[i]} &= K_1^{[i]} + K_2^{[i]} \end{aligned}\right\} \quad \textbf{for } i = 1, \dots, m_u$$

Generalized observer for sensor faults using UIO

Each residual generator of this type is **driven by all outputs and all but one inputs**

When all **sensors are fault-free** and a **fault occurs in the** i-th **actuator**, the residuals will satisfy the following isolation logic:

$$\begin{cases} \left\|\boldsymbol{r}^{[i]}(t)\right\| < \eta_i^{\text{actuator}} \\ \left\|\boldsymbol{r}^{[k]}(t)\right\| \geq \eta_k^{\text{actuator}} \quad \text{for} \quad k = 1, \dots, i-1, i+1, \dots, p \end{cases}$$

Example: actuator fault isolation with UIO

Consider the following MIMO system with $m_u = m_{u_a} = 2, p = 3, m_f = 1$, sampled at $T_s = 1$ s.

Assume there are no disturbances

$$\underset{n\times n}{A} = \underset{4\times 4}{\begin{bmatrix} 0.5 & -0.7 & 0.7 & 0 \\ 0 & 0.8 & 0.06 & 0 \\ -1 & 0 & 0 & 0.1 \\ 0 & 0 & -0.1 & 0.4 \end{bmatrix}} \quad \underset{n\times m_u}{B} = \underset{4\times 2}{\begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}} \quad \underset{p\times n}{C} = \underset{3\times 4}{\begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}} \quad \underset{p\times m_u}{D} = \underset{3\times 2}{\begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}}$$

$$\underset{n\times m_f}{B_f^{[1]}} = \underset{4\times 1}{\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}} \quad \underset{n\times m_f}{B_f^{[2]}} = \underset{4\times 1}{\begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}} \quad \underset{p\times m_f}{D_f} = \underset{3\times 1}{\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}}$$

- Observers poles: 0.0729, 0.197, 0.0053, 0.27
- $\boldsymbol{u}(t) = \boldsymbol{u}_a(t) \sim \text{WN}(0, I_2)$

Collect $N = 200$ data. Simulate a step-like actuator fault of amplitude 1 which acts at time $t = 100$ s, one time on the first input $u_1(t)$ and then on the second input $u_2(t)$

Example: actuator fault isolation with UIO

We build two UIO using the generalized observer scheme:

1. The first UIO is insensitive to faults on $u_{a,1}(t)$, and it is driven only by $u_{a,2}(t)$
2. The second UIO is insensitive to faults on $u_{a,2}(t)$, and it is driven only by $u_{a,1}(t)$

Define the quantities

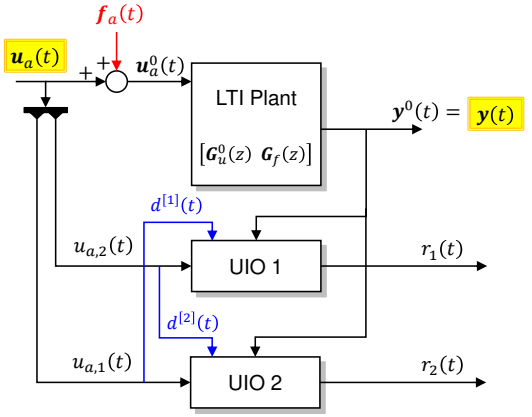
$$B_d^{[1]} = [B_d \;\; \boldsymbol{b}_1] = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \qquad B_d^{[2]} = [B_d \;\; \boldsymbol{b}_2] = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

$$\boldsymbol{d}^{[1]}(t) = \begin{bmatrix} \boldsymbol{d}(t) \\ u_{a,1}(t) + f_{a,1}(t) \end{bmatrix} = [u_{a,1}(t) + f_{a,1}(t)] \qquad \boldsymbol{d}^{[2]}(t) = \begin{bmatrix} \boldsymbol{d}(t) \\ u_{a,2}(t) + f_{a,2}(t) \end{bmatrix} = [u_{a,2}(t) + f_{a,2}(t)]$$

Example: actuator fault isolation with UIO

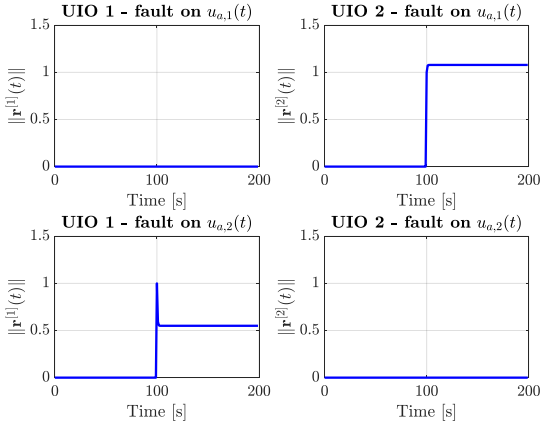
The diagnostic configuration of the considered example

- **UIO 1** is fed by $u_{a,2}(t)$ and $\boldsymbol{y}(t)$. The input $u_{a,1}(t)$ is treated as a disturbance
- **UIO 2** is fed by $u_{a,1}(t)$ and $\boldsymbol{y}(t)$. The input $u_{a,2}(t)$ is treated as a disturbance



Example: actuator fault isolation with UIO

- The residual $r_1(t)$ is sensitive to faults in $u_{a,2}(t)$
- The residual $r_2(t)$ is sensitive to faults in $u_{a,1}(t)$



UNIVERSITÀ DEGLI STUDI DI BERGAMO | Dipartimento di Ingegneria Gestionale, dell'Informazione e della Produzione

UNIVERSITÀ DEGLI STUDI DI BERGAMO | Dipartimento di Ingegneria Gestionale, dell'Informazione e della Produzione

ADAPTIVE LEARNING, ESTIMATION AND SUPERVISION OF DYNAMICAL SYSTEMS (ALES)

Lecture 16: Signal-based fault diagnosis

Master Degree in COMPUTER ENGINEERING

Data Science and Data Engineering Curriculum

SPEAKER
Prof. Mirko Mazzoleni

PLACE
University of Bergamo

Syllabus

1. **Recursive and adaptive identification**
   - 1.1 Recursive ARX estimation (RLS)
   - 1.2 Least Mean Squares (LMS)
   - 1.3 Instrumental Variables (IV)
2. **Closed-loop identification**
3. **Subspace and MIMO identification**
   - 3.1 Singular Value Decomposition
   - 3.2 Impulse data: Ho-Kalman, Kung algorithms
   - 3.3 Generic I/O data: the MOESP algorithm
4. **Supervision of dynamical systems**
   - 4.1 Introduction to fault diagnosis
   - 4.2 Model-based fault diagnosis
   - 4.3 Parity space approaches
   - 4.4 Observer-based approaches
   - 4.5 Signal-based fault diagnosis
   - 4.6 Knowledge-based fault diagnosis

CASE STUDIES

- Virtual Reference Feedback Tuning
- Nuclear particles classification
- Leak detection in an industrial valve
- Bearing fault identification

Outline

1. Review of the signal-based fault diagnosis rationale
2. Vibration analysis of rotating components
3. Ball bearings diagnosis
4. Gears diagnosis
5. Order analysis

Outline

1. **Review of the signal-based fault diagnosis rationale**
2. Vibration analysis of rotating components
3. Ball bearings diagnosis
4. Gears diagnosis
5. Order analysis

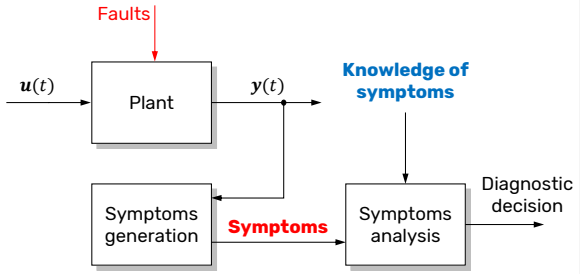
Signal-based fault diagnosis

Certain process signals carry information about the faults to be detected

- From them, **fault symptoms** are computed…
- …and compared with **prior knowledge about the faults**

Examples are:

- Diagnosis of rotating components defects
- Stator\rotor current faults



Signal-based fault diagnosis

Fault symptoms can be extracted from signals using:

1. **Time** domain methods, *RMS, **kurtosis**, **crest factor**, zero-crossings, correlation, envelope analysis, Dynamic Time Warping (DTW), cycle counting (velocity reversals)*
2. **Frequency** domain methods, ***FFT-based**, order analysis, cepstrum, Mel Freq. Cepstrum (MFCC)*
3. **Time-Frequency** domain methods, ***Short-time Fourier Transform (STFT),** spectral kurtosis, wavelet transform, Empirical Mode Decomposition (EMD), Wigner-Ville distribution*

Outline

1. Review of the signal-based fault diagnosis rationale
2. **Vibration analysis of rotating components**
3. Ball bearings diagnosis
4. Gears diagnosis
5. Order analysis

Vibration analysis of rotating components

Many machines contain drive systems with *motors*, *clutches*, *gears*, *shafts*, *belts* or chains, and different ball/rolling or oil *bearings*

Vibrations are usually generated by:

- Inherent **machine oscillations** (e.g. piston-crankshaft, toothed machine tool cutting, axial piston pumps, electromechanical motors)
- **Shaft oscillations** with radial or axial displacement
- **Irregular speed** of the shaft
- **Impuse-wise excitation**, though e.g. backlash, cracks, pittings, broken gear teeth

Vibration analysis of rotating components

Some vibrations indicate a **normal state** of the machines. However, **changes** of these vibrations and **additional appearing** ones may be caused by faults

Examples of machinery defects that can be detected using vibrations analysis are:

- Unbalance
- Bent shaft
- Eccentricity
- Misalignment
- Rotor rubs
- Resonance
- Belt drive problems
- Gear defects
- Bearing defects
- Electrical faults
- Cavitation
- Shaft cracks
- Looseness
- Oil whip/whirl
- Hydraulic and aerodynamic forces

Vibration analysis of rotating components

Distinguish all frequency components is more difficult in time domain. **Vibration analysis** therefore usually is performed in **frequency** or **time-frequency** domain

Machine vibrations are usually measured as **acceleration** by accelerometers at the casing of the machines

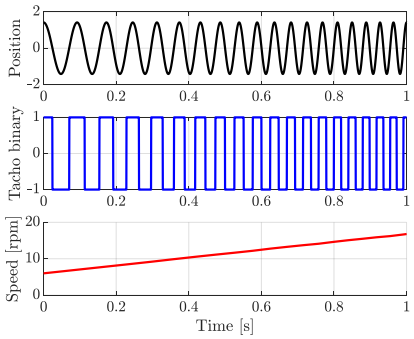
Thus, there is a **machine transfer behaviour** (e.g. several **resonance** frequencies of the machine structure) between the source of the vibrations and the measurement location

Tachometric signal

A **tachometric signal** provides an indication of the **rotation speed** by an *Hall sensor or optical encoder*, and it is composed by pulses where the time between the pulses is related to the rotation speed

$$\omega(t) = \frac{60\ \text{s/min}}{N_p(t_2 - t_1)\ \text{s}} \quad [\text{rpm}]$$

- N_p : number of sensor pulses per revolution
- t_1, t_2: time instants of the two pulses, in seconds



RPM maps

A typical analysis in rotating machinery analysis is to make a **run-up** *(from low to high rpm)* or a **coast-down** *(from high to low or zero rpm, i.e. switching off)* of the machine, measuring vibrations

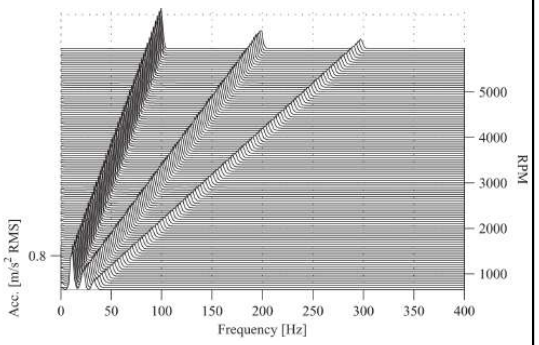
The vibration signal is then divided into **short segments**, for each of which an instantaneous spectrum is calculated (Short-time Fourier Transform, or **STFT**)

An **RPM map** is a representation of a spectrum as function of the rpm speed

Waterfall plots

The **waterfall plot** is a way for representing an RPM map:

- spectrum components that are **proportional to rotation speed** will be visible as peaks on a **straight line**
- fixed (not depending on velocity) **structural resonances** will be visible as peaks at **fixed frequencies**



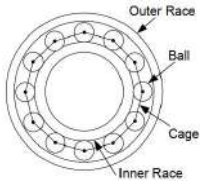
Outline

1. Review of the signal-based fault diagnosis rationale
2. Vibration analysis of rotating components
3. **Ball bearings diagnosis**
4. Gears diagnosis
5. Order analysis

Mechanical bearings

They are composed by several components:

- **Races:** the surfaces on which the bearing rolls. The load placed on the bearing is supported by this contact surface. In general, the inner ring rests on the shaft, while the outer ring rests on the bearing housing
- **Rolling elements:** the rolling elements are constructed in such a way as to allow also their rotation (simultaneous rotation around their axis and around the axis of the bearing)



- **Cage:** separates the rolling elements at a regular interval, holds them in position between the internal and external races and allows their free rotation

Rolling bearings defects

Analytical formulas exist to describe the **base frequencies excited** by localized defects

BallPass Frequency Inner race

Fault on the inner race

$$\mathrm{BPFI} = \frac{n f_r}{2}\left\{1 + \frac{d}{D}\cos\phi\right\}$$

BallPass Frequency Outer race

Fault on the outer race

$$\mathrm{BPFO} = \frac{n f_r}{2}\left\{1 - \frac{d}{D}\cos\phi\right\}$$

Ball (roller) Speed Frequency

Fault on rolling elements

$$\mathrm{BSF(RSF)} = \frac{D}{2d} f_r \left\{1 - \left(\frac{d}{D}\cos\phi\right)^2\right\}$$

Fundamental Train Frequency

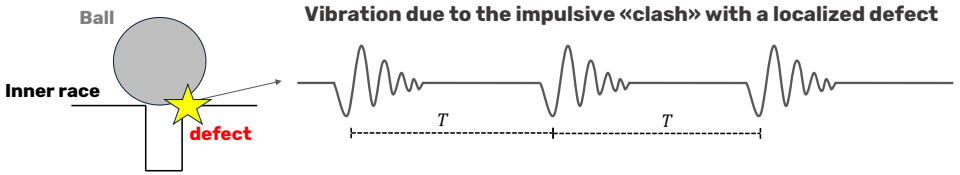
Fault on cage

$$\mathrm{FTF} = \frac{f_r}{2}\left\{1 - \frac{d}{D}\cos\phi\right\}$$

- f_r: shaft speed
- ϕ: angle of the load from radial plane
- d: single rolling element diameter
- D: average bearing diameter (pitch)
- n: number of rolling elements of the bearing

Rolling bearings diagnosis

Local faults in rolling element bearings produce a **series of impacts** which repeat (almost) **periodically** at a rate dependent on bearing **geometry** and **rotation speed**



The **diagnostic information** of interest is contained in the **repetition frequency of the impact series**, and not in the frequency spectrum resulting from the impacts, as this would usually be a sum of the resonance frequencies excited

Rolling bearings diagnosis

Due to the **nature of the fault**, the best way to detect fault-generated **vibrations** is to use an **accelerometer**

However, vibration signals are often **severely corrupted** by strong levels of background noise, encompassing all **other vibration sources** in the machine under inspection

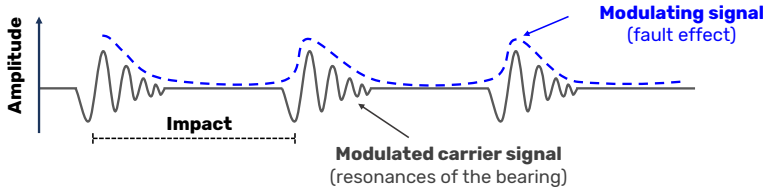
Frequency analysis based on the FFT **on the raw vibration** signal **is not useful** due to:

1. The before mentioned noise
2. Random **fluctuations in the shaft speed**, which compromise the repeatability of the fault impulses responses *(if the speed changes, the fault frequencies change too)*

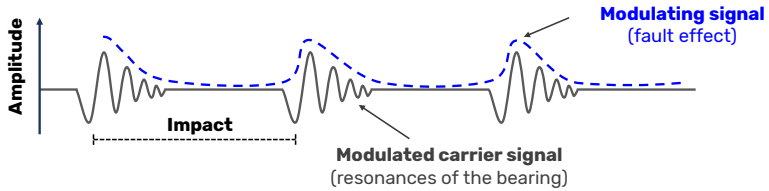
Envelope analysis

One way to enhance the fault frequencies is by using the **envelope analysis**

As we saw, **impactive faults** excite the **structural resonances** of bearing, simply **amplifying** standard operational vibrations. This variation effect on the amplitude of the natural frequency is known as **amplitude modulation**



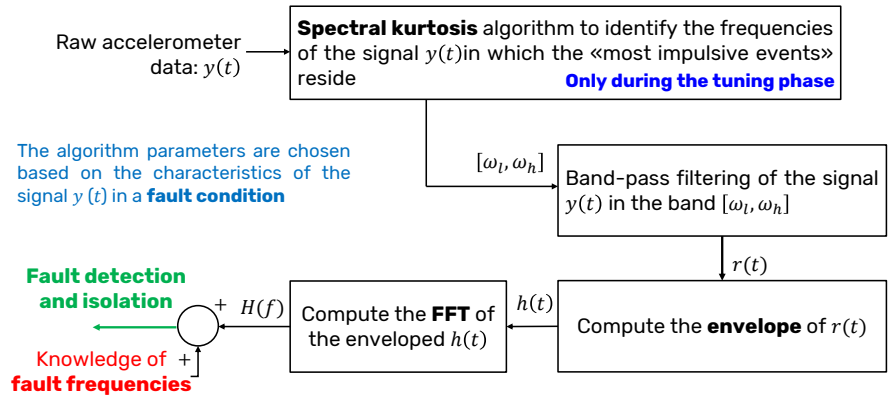
Envelope analysis



The aim of envelope analysis (also known as amplitude demodulation) is to **reconstruct** the **modulating signal** from the measured **modulated signal**

Then, a **frequency analysis** of the resulting modulating signal can be performed to **evaluate the presence** of the fault frequency (and its harmonics)

General scheme for bearing diagnosis



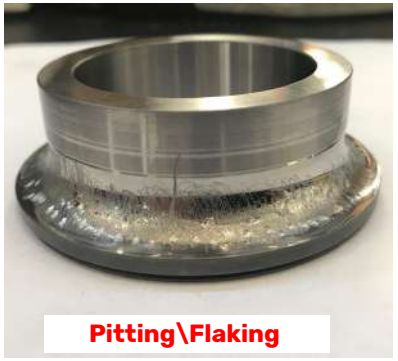
Example: bearing diagnosis for workcenters

AIM: monitoring the **bearing** of a control center (CNC machine) vertical shaft

This kind of machine is use for **highly precision** manufacturing **operations**

These centers can work on the three Euclidean axes, and also on several rotational ones (where the **spindle** is oriented)

Example: bearing diagnosis for workcenters



Example: bearing diagnosis for workcenters

Accelerometer on bearing housing

Chassis NI CompactDAQ 9184

NI 9230 module

For the development of the algorithm for the diagnosis of the bearing, the Y axis of the accelerometer is considered.

Sampling frequency: 12800 Hz

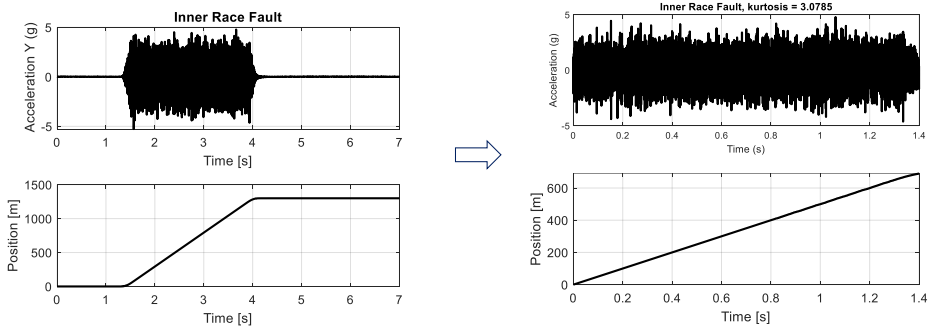
Acceralation range: ± 50 g

Frequency range: $\pm[0.5 - 3000]$Hz

Piezoelectric accelerometer

Step 1: constant speed data

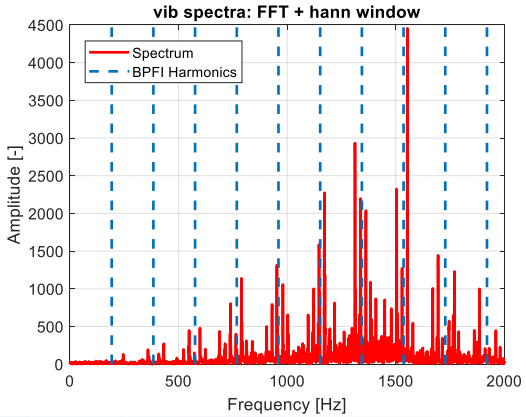
It is necessary to maintain only the vibration signal section that corresponds to a **constant rotational speed** of the shaft



Application of the FFT to raw data

The direct application of the FFT to the signal $y(t)$, at constant speed, **does not lead** to immediate recognition of the fault

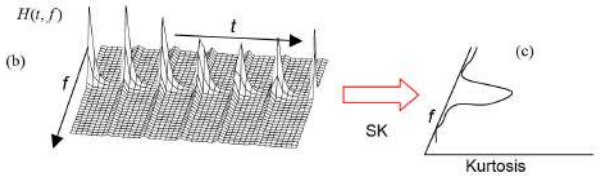
Several **signal preprocessing** stages are required to bring out the fault symptoms



Step 2: spectral kurtosis

The **spectral kurtosis** algorithm is used to select the frequency bands that exhibit «more impulsivity» or «energy». The goal is to **isolate the impulsive fault components**, by filtering the signal in the frequency bands found by the SK algorithm

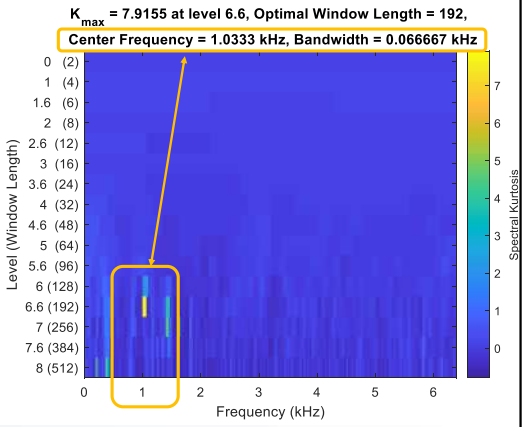
Spectral kurtosis (SK) is obtained by computing the **kurtosis for each frequency** (or frequency window) in a **time-frequency diagram** (spectrogram)



The MatLab `kurtogram` algorithm computes the SK at different «frequency resolutions», then selecting the one with the highest kurtosis value

Spectral kurtosis

- By applying the SK algorithm to the signal $y(t)$, we obtain a graph showing the kurtosys value as the **frequency** and **frequency resolution** (level) vary
- The algorithm also returns an optimal level of **frequency band** in which to filter
- These values should be taken only as an **indicative suggestion** for a correct selection of the filter band. The chosen value $[\omega_l, \omega_h]$ is then **fixed for all tests**

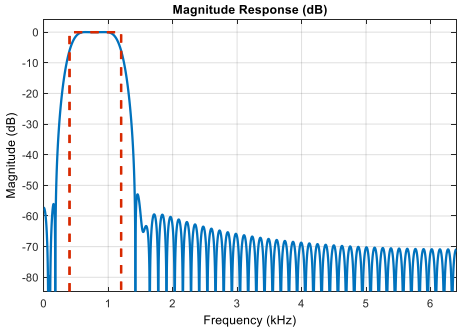


Step 3: band-pass filtering

The $y(t)$ signal is **band-pass filtered** using the values $[\omega_l, \omega_h]$ selected with the help of the SK, obtaining the $r(t)$ signal

It is possible, for simplicity, to use a **FIR filter** of an appropriate order based on the required level of attenuation

In the example, we chose a FIR filter of order 100 centered at $f_c = 800\ \text{Hz}$ and bandwith $b = 800\ \text{Hz}$



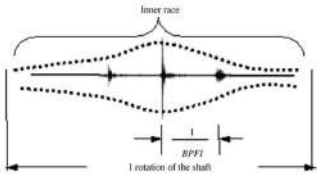
Step 4: envelope analysis

Compute the **envelope** $h(t)$ of the filtered band-pass signal $r(r)$

In the case of an **inner race fault**, the accelerometric signal is modulated by the **rotation speed** of the shaft (which runs at the same speed as the inner race)

The fault information is contained in the **«time» between the pulses** $\left(\frac{1}{\text{BPFI}}\text{s}\right)$ rather than in their frequency content

The computation of the signal envelope allows to be **more robust** in the evaluation of the «time» between pulses



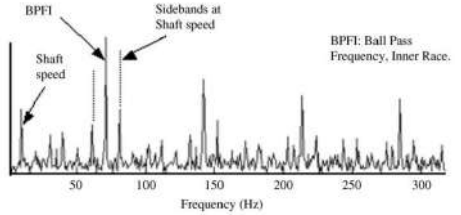
Accelerometric signal in the presence of an inner race fault

Step 5: FFT of the envelope

The **FFT** $H(f)$ of the **envelope** $h(t)$ is computed to diagnose the fault

The aim is to observe the presence of a **high amplitude value** in the **typical fault frequencies** (BPFI and multiples), and to evaluate its energy compared to the case of a healthy bearing

It is very probable, in the case of an inner race fault, to observe a **modulation** due to the frequency of rotation of the bearing shaft (and the inner race) at the frequencies of the fault



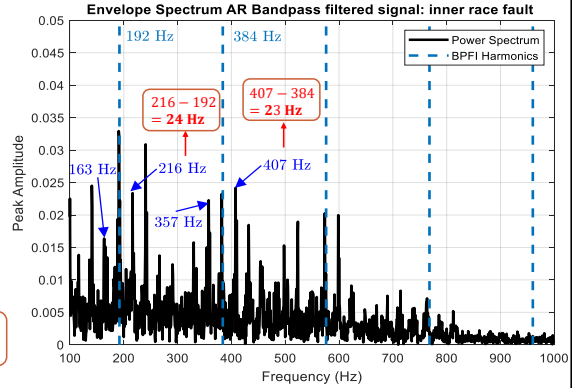
Results

BPFI fault frequencies are clearly visible

Note the **modulation of the BPFI** frequencies with the **rotation frequency** of the bearing

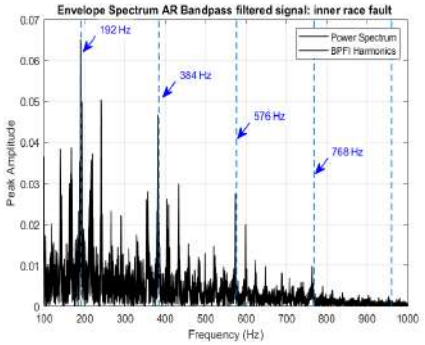
$$\omega = \frac{30000\ \text{mm/min}}{20\ \text{mm/round}} = 1500\ \text{round/min}$$

$$f_m = \frac{\omega}{60\ \text{s/min}} = \frac{1500\ \text{round/min}}{60\ \text{s/min}} = \mathbf{25\ Hz}$$

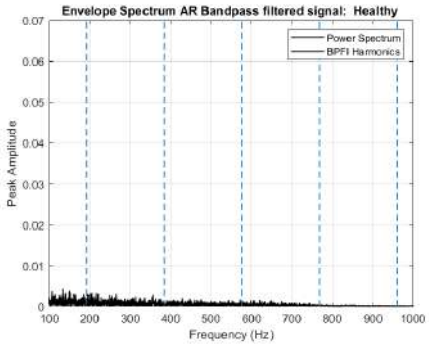


Results

Inner race fault bearing



Healthy bearing



Outline

1. Review of the signal-based fault diagnosis rationale
2. Vibration analysis of rotating components
3. Ball bearings diagnosis
4. **Gears diagnosis**
5. Order analysis

Mechanical gears

Gears are very widely used in machines to **transmit power** from one shaft to another, usually with a **change in speed and torque**

A gearbox is a piece of rotating equipment that can cause the normal **low-frequency** harmonics in the vibration spectrum, but also might show a lot of activity in the **high-frequency** region due to **gear teeth** and bearing **impacts**

The spectrum of any gearbox shows the $1\times$ and $2\times$ rpm, along with the **gear mesh frequency** (GMF):

$$\text{GMF} = \text{number of teeth on pinion} \times \text{pinion rpm}$$

Here the GMF is expressed in rpm and not in Hz

Mechanical gears

The GMF will have **running speed sidebands** relative to the shaft speed to which the gear is attached. These contain information about **gearbox faults** *(e.g. tooth wear, backlash)*

As a general rule:

- **distributed faults** such as *eccentricity* and *gear misalignment* will produce sidebands and harmonics that have **high amplitude close to the gear-mesh frequency**
- **localized faults** such as a *cracked tooth* produce sidebands that are **spread more widely** across the spectrum

Mechanical gears

The **gear hunting tooth frequency** indicates the *frequency with which a tooth of the gear engages with the same tooth of the pinion (another gear).* It is particularly effective for detecting faults on both the gear and the pinion

It can cause quite high vibrations, but since it occurs at **low frequencies**, predominantly less than 600 rpm, it is often missed during vibration analysis

$$\text{Hunting Tooth Frequency} = \frac{\text{GMF} \cdot N_A}{(\text{no. of pinion teeth}) \cdot (\text{no. of gear teeth})}$$

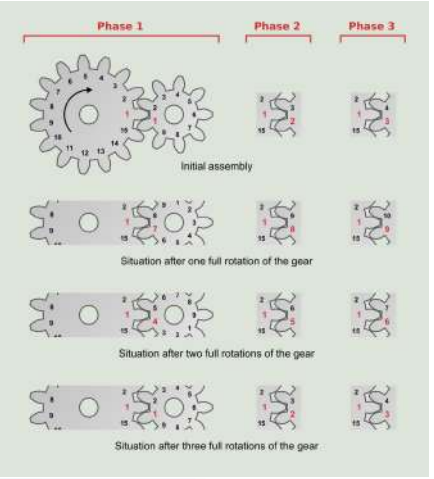
where N_A is the **assembly phase factor**, i.e. the *greatest common divisor* (usually a low number) between the number of teeth on the pinion and gear

Mechanical gears

The **assembly phase factor** is defined as the different **modes of engagement** between a pair of gears

Consider a gear with 15 teeth and one with 9. Then, $N_A = 3$: there are three different ways of assembling the gear. This means that there are **three possible wear patterns** when engaging the gear and pinion

It is always best to have $N_A = 1$, so that each gear tooth eventually mates with every pinion tooth



Time-Synchronous Averaging (TSA)

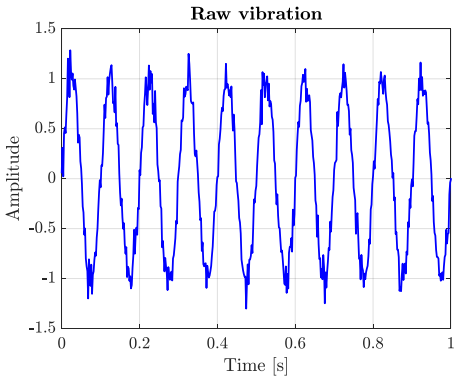
Time-Synchronous Averaging resamples the vibration data *synchronously* with shaft **rotation angles**. Then, average is performed «rotation-wise» rather than «time-wise»

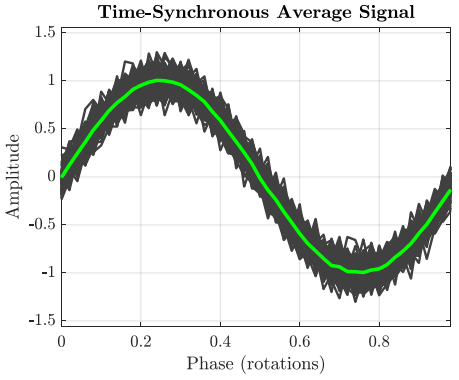
Averaging over rotation angles **aid to reject** any noise, disturbance, or periodic signal content that is **not coherent with the rotation**

Thus, a **tachometric signal** is needed to measure the **time instants** when a full rotation is completed

Then, proper **interpolation algorithms** are used to resample the signal with respect to the rotation speed

Time-Synchronous Averaging (TSA)





https://it.mathworks.com/help/signal/ref/tsa.html

Time-Synchronous Averaging (TSA)

TSA is well suited for gearbox analysis, where it allows the **vibration signature** of the gear under analysis to be separated from other gears and noise sources in the gearbox that are **not synchronous** with that gear

Additionally, **variations in shaft speed** can be corrected, thus reducing the frequency spreading in an FFT analysis

Three main filtered versions of the TSA signal are derived for further analysis:

1. **Residual signal**
2. **Difference signal**
3. **Regular signal**

Residual signal

The **residual signal** is computed from the TSA signal by removing the following frequencies from the signal spectrum

- **Shaft frequency** and its **harmonics**
- **Gear meshing frequencies** and their **harmonics**

The frequencies are removed by computing the discrete Fourier transform (DFT) and **setting the spectrum values to zero** at the specified frequencies

Difference signal

The difference signal is computed from the TSA signal $y_{\text{raw}}(t)$ by filtering the following from the signal spectrum:

- **Shaft frequency** and its **harmonics**
- **Gear meshing frequencies** and their **harmonics**
- **First-order sidebands** at the gear meshing frequencies and their harmonics

Regular signal

Given a raw signal $y_{\text{raw}}(t)$ The **regular signal** $y_{\text{reg}}(t)$ is related to the residual signal $y_{\text{res}}(t)$ through the equation

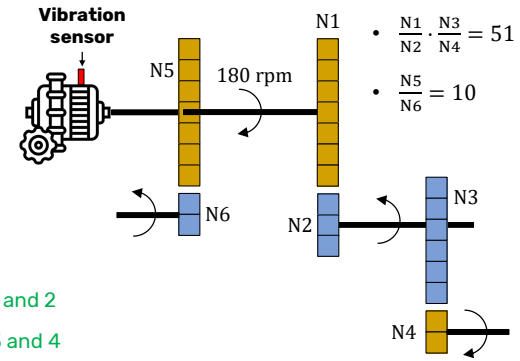
$$y_{\text{reg}}(t) = y_{\text{raw}}(t) - y_{\text{res}}(t)$$

If the first-order sidebands are retained in the regular signal, then:

$$y_{\text{reg}}(t) = y_{\text{raw}}(t) - y_{\text{diff}}(t)$$

Example: gears vibration signals

Consider a drivetrain with six gears driven by a motor that is equipped with a vibration sensor



- $\frac{N1}{N2} = 17$
- $\frac{N1}{N2} \cdot \frac{N3}{N4} = 51$
- $\frac{N5}{N6} = 10$

The main frequencies that appear in the measured vibration signal are:

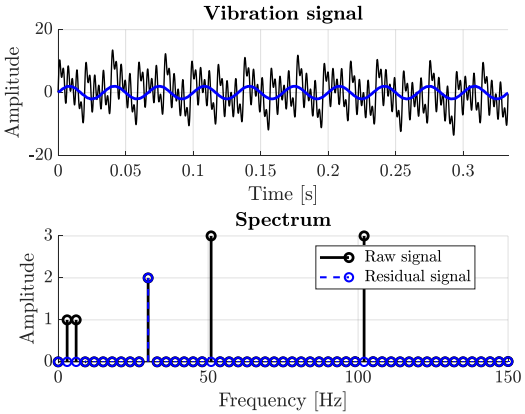
- f_1: $\frac{180}{60} = 3$ Hz
- $f_{12} = f_1 \cdot 17 = 51$ Hz GMF for gears 1 and 2
- $f_{34} = f_1 \cdot 51 = 153$ Hz GMF for gears 3 and 4
- $f_{56} = f_1 \cdot 10 = 30$ Hz GMF for gears 5 and 6

Example: gear vibration signals

Define a signal as

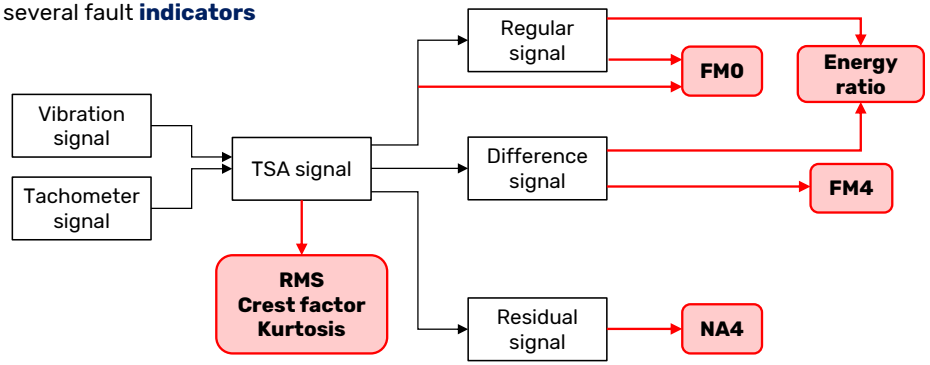
$$y(t) = \sin(2\pi f_1 \cdot t) + \sin(2\pi 2 f_1 \cdot t) + 3\sin(2\pi f_{12} \cdot t) + 3\sin(2\pi 2 f_{12} \cdot t) + 4\sin(2\pi f_{34} \cdot t) + 4\sin(2\pi 2 f_{34} \cdot t) + 2\sin(2\pi f_{56} \cdot t)$$

Assume we are interested only in the vibration signal produced by gears 5 and 6



Fault indicators for gears

Using the TSA, residual, difference and regular gears signals, it is possible to compute several fault **indicators**



Fault indicators for gears

- **RMS:** good indicator of the overall condition of gearboxes and for detecting unbalanced rotating elements, but not a good indicator of incipient tooth failure since it is not sensitive to peaks in the vibration signal

$$\mathrm{RMS}(y_{\mathrm{raw}}(t)) = \sqrt{\frac{1}{N}\sum_{t=1}^{N} y_{\mathrm{raw}}(t)^2}$$

- **Kurtosis:** captures the amount of peaks in the vibration signals, which we may assumed are due to a damage

$$\mathrm{Kurtosis} = \frac{\frac{1}{N}\sum_{t=1}^{N}(y_{\mathrm{raw}}(t) - \bar{y}_{\mathrm{raw}})^4}{\left(\frac{1}{N}\sum_{t=1}^{N}(y_{\mathrm{raw}}(t) - \bar{y}_{\mathrm{raw}})^2\right)^2}$$

- $\bar{y}_{\mathrm{raw}}(t)$: average value of $y_{\mathrm{raw}}(t)$

Fault indicators for gears

- **Crest factor:** indicates a damage in an early stage. When only one tooth is damaged, there is no change in the RMS value of the vibration signal during one rotation of the drive shaft where the damaged gear is located, while the peak value increases

$$\text{Crest factor} = \frac{\text{peak}\left(y_{\text{raw}}(t)\right)}{\text{RMS}\left(y_{\text{raw}}(t)\right)}$$

- **FM4:** kurtosis of the difference signal. Detects faults isolated to only a finite number of teeth

$$\text{FM4} = \frac{\frac{1}{N}\sum_{t=1}^{N}\left(y_{\text{diff}}(t) - \bar{y}_{\text{diff}}\right)^4}{\left(\frac{1}{N}\sum_{t=1}^{N}\left(y_{\text{diff}}(t) - \bar{y}_{\text{diff}}\right)^2\right)^2}$$

Fault indicators for gears

- **FM0:** identifies major anomalies, such as tooth breakage or heavy wear, in the meshing pattern of a gear

$$\text{FM0} = \frac{\text{peak}\left(y_{\text{raw}}(t)\right)}{\text{RMS}\left(y_{\text{regular}}(t)\right)}$$

- **Energy ratio:** indicates heavy wear, where multiple teeth on the gear are damaged

$$\text{Energy ratio} = \frac{\text{standard deviation}\left(y_{\text{diff}}(t)\right)}{\text{standard deviation}\left(y_{\text{reg}}(t)\right)}$$

Fault indicators for gears

- **NA4:** An improved version of the FM4 indicator. The NA4 uses the residual signal and a current average variance over the last M portions of the residual signal.

It indicates the onset of damage and continues to react to the damage as it spreads and increases in magnitude

$$\text{FM4} = \frac{\frac{1}{N}\sum_{t=1}^{N}\left(y_{\text{res}}(t) - \bar{y}_{\text{res}}\right)^4}{\left(\frac{1}{M}\sum_{m=1}^{M}\frac{1}{N}\sum_{t=1}^{N}\left(y_{\text{res},m}(t) - \bar{y}_{\text{res},m}\right)^2\right)^2}$$

where $y_{\text{res},m}(t)$ indicates the m-th portion of N data of the residual signal

Outline

Order analysis

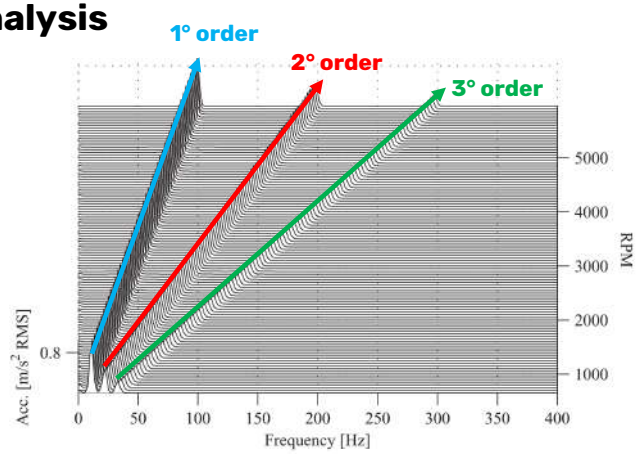
Frequency analysis methods based on the FFT and envelope signal requires that the **rotation speed** is **constant**

In fact, the typical **fault frequencies** depend on the **rotation speed**: if the speed change, the fault frequencies change too! *In practice, there are always fluctuations of the speed*

Thus, if we perform a frequency analysis of a vibration signal associated to **varying rotation speeds**, the fault frequencies get **«smeared»** to a wider portion of the spectrum

Order analysis is like a frequency analysis, «normalized» with respect to the rotation speed, thus allowing an analysis also with varying speed

Order analysis



Order analysis

Order analysis requires a **tachometric signal** or a measurement of the rotation speed

$$\text{Rotational speed (rpm)} = \text{first order}$$

A **multiple** n of the main rotation speed is the n-th **order**

Like for TSA, the vibration signal is **resampled** so that the sampling is like as it was performed **«revolution-wise»** instead of «time-wise». This is also called **synchronous sampling**

Order analysis

The aim of **synchronous sampling** is to sample the vibration or noise signal at **equal angles** along each cycle (one revolution being one cycle), instead of equidistantly in time

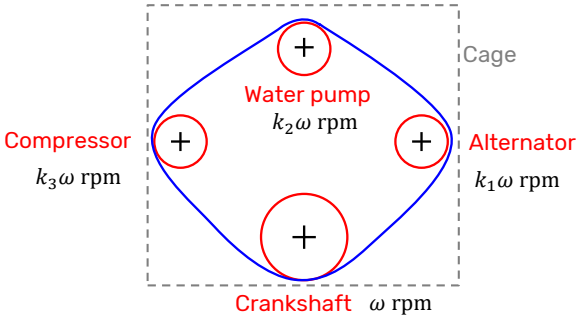
The samples after synchronous sampling are usually said to be in the **angle domain.** In the angle domain, the signals will have **equally spaced samples** with respect to the **rotational angle**

By performing an FFT of the angle domain signal, we get an **order spectrum**, where the problem of **frequency smearing is avoided**

It is the possible to **track the magnitude** of certain orders to perform **monitoring**

Order analysis

Consider an engine. Order analysis helps to identify **frequency components that change with speed** from frequency components that do not change with speed

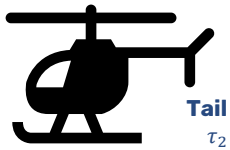


The natural frequencies of the **cage** do not depend on the motor speed

Example: helicopter vibration

A helicopter has several rotating components, including the **engine**, **gearbox**, and the **main** and **tail rotors**

Main rotor transmission ratio $\tau_1 = 0.0129$



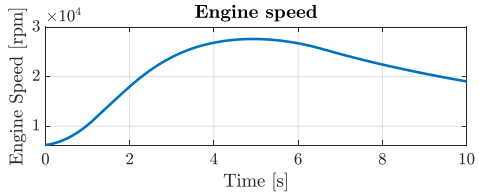
Tail rotor ratio $\tau_2 = 0.0658$

Consider an helicopter with **4 blades** in both the main and tail rotors. **Vibration data** are measured in the helicopter **cabin**

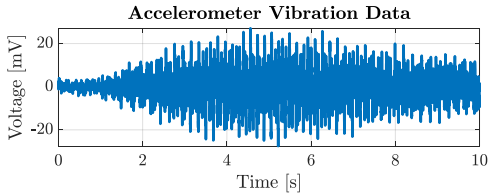
The frequency of dominant vibration components can be related to the rotational speed of the motor to **investigate the source of high-amplitude vibration**

Example: helicopter vibration

The **engine speed** increases during the run-up and decreases during the coast-down



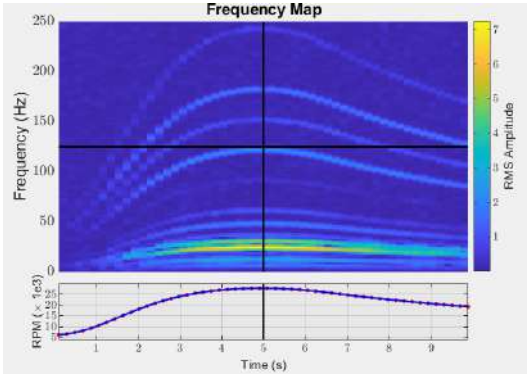
The **vibration amplitude** changes as a function of rotational speed



Example: helicopter vibration

Use a **STFT** to generate an **RPM-frequency map**

Many of the tracks in the RPM-frequency maps have frequencies that increase and decrease with the motor speed. This suggests that the tracks are **orders** of the motor rotational frequency.

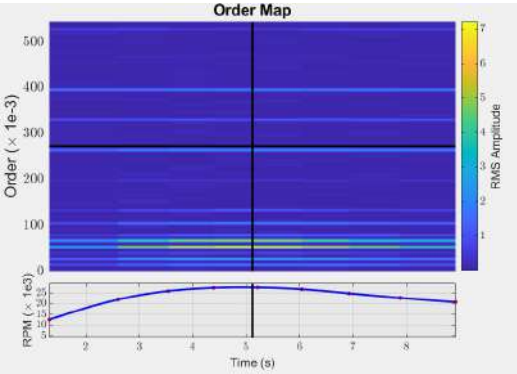


Example: helicopter vibration

Use a **STFT** to generate an **RPM-order map**

The map contains a **straight track** for each **order**, indicating that the vibration occurs at a fixed multiple of the motor rotational speed

Smearing artifacts are **reduced**



Example: helicopter vibration

In the RMP-order map:

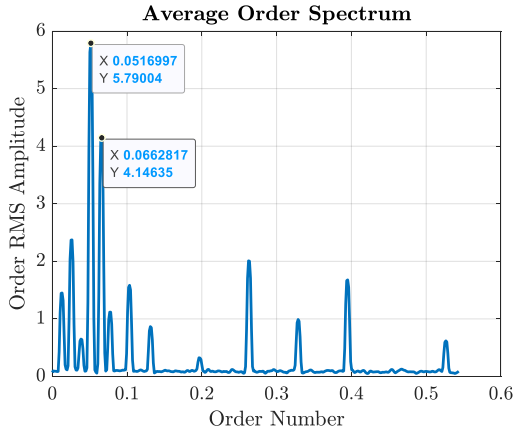
- Frequencies **dependent** on the **speed** are shown as **horizontal lines**
- Frequencies that **do not depend** on the **speed** will be shown as **curves**

Example: helicopter vibration

Compute a FFT spectrum and locate the **orders** where there are **peaks** to get the orders where vibrations are highest

The highest amplitude orders are:

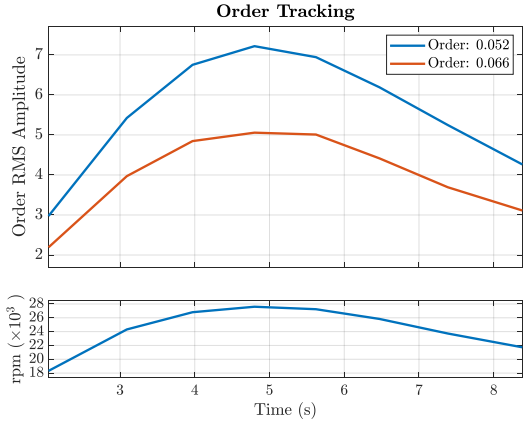
- $o_1 = 0.052$
- $o_2 = 0.066$



Example: helicopter vibration

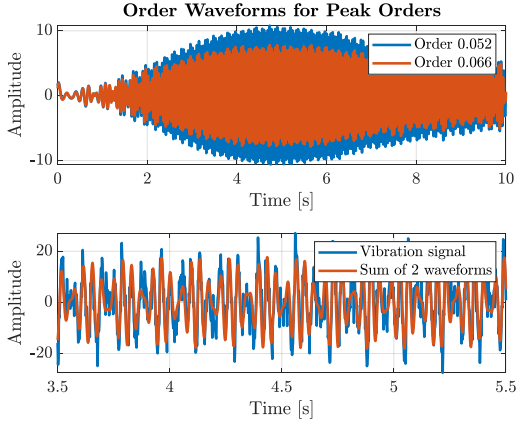
Represent the **order RMS over time**

Both orders increase in amplitude as the **rotational speed** of the motor increases



Example: helicopter vibration

It is possible to estimate the **waveforms** due to the **most important orders** and compare them with the overall vibration signal



Example: helicopter vibration

We can relate the highest amplitude orders with the transmission ratio of the main rotor and tail rotor

$$\frac{o_1}{\tau_1} = \frac{0.052}{0.0129} = 4.031 \qquad \frac{o_1}{\tau_2} = \frac{0.052}{0.0658} = 0.7903$$

$$\frac{o_2}{\tau_1} = \frac{0.066}{0.0129} = 5.116 \qquad \frac{o_2}{\tau_2} = \frac{0.066}{0.0658} = 1.003$$

We can conclude that:

- o_1 might be related to the main rotor (4 blades)
- o_2 might be related to the tail motor

⇨ Perform **corrective actions** on these two components!

UNIVERSITÀ DEGLI STUDI DI BERGAMO | Dipartimento di Ingegneria Gestionale, dell'Informazione e della Produzione

ADAPTIVE LEARNING, ESTIMATION AND SUPERVISION OF DYNAMICAL SYSTEMS (ALES)

Lecture 17: Knowledge-based fault diagnosis

Master Degree in COMPUTER ENGINEERING

Data Science and Data Engineering Curriculum

SPEAKER
Prof. Mirko Mazzoleni

PLACE
University of Bergamo

Syllabus

1. **Recursive and adaptive identification**
   - 1.1 Recursive ARX estimation (RLS)
   - 1.2 Least Mean Squares (LMS)
   - 1.3 Instrumental Variables (IV)
2. **Closed-loop identification**
3. **Subspace and MIMO identification**
   - 3.1 Singular Value Decomposition
   - 3.2 Impulse data: Ho-Kalman, Kung algorithms
   - 3.3 Generic I/O data: the MOESP algorithm
4. **Supervision of dynamical systems**
   - 4.1 Introduction to fault diagnosis
   - 4.2 Model-based fault diagnosis
   - 4.3 Parity space approaches
   - 4.4 Observer-based approaches
   - 4.5 Signal-based fault diagnosis
   - 4.6 Knowledge-based fault diagnosis

CASE STUDIES

- Virtual Reference Feedback Tuning
- Nuclear particles classification
- Leak detection in an industrial valve
- Bearing fault identification

Outline

1. Introduction to Principal Component Analysis (PCA)
2. PCA problem formulation
3. Choice of the number of components
4. PCA-based process monitoring indicators

Outline

1. **Introduction to Principal Component Analysis (PCA)**
2. PCA problem formulation
3. Choice of the number of components
4. PCA-based process monitoring indicators

Principal Component Analysis: motivation

Principal Component Analysis (PCA) can be used for different purposes:

- **Data compression:** project the data to a lower dimensionality (from $\mathbb{R}^d$ to $\mathbb{R}^q$, $q < d$)
- **Data visualization:** visualize in a 2-D plot high-dimensional data

PCA produces a **low-dimensional representation** of a dataset:

- it finds **linear combinations** of the original features that have maximal variance
- the derived variables are **mutually uncorrelated**
- the derived variables can be used in supervised learning problems to **speed up** the computation

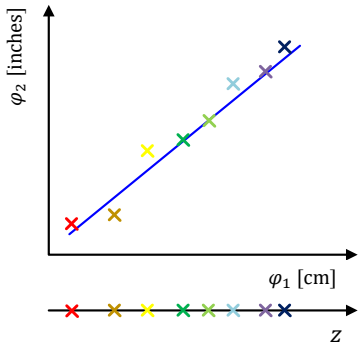
Principal Component Analysis: motivation

Reduce the data dimensionality from $d = 2$ to $q = 1$

- Highly **redundant** representation
- The compressed data retain the **main information**
- Approximation but **reduced space** requirement

$$\boldsymbol{\varphi}(1) \in \mathbb{R}^2 \rightarrow z(1) \in \mathbb{R}$$
$$\boldsymbol{\varphi}(2) \in \mathbb{R}^2 \rightarrow z(2) \in \mathbb{R}$$
$$\vdots$$
$$\boldsymbol{\varphi}(N) \in \mathbb{R}^2 \rightarrow z(N) \in \mathbb{R}$$

φ_1 and φ_2 are not perfectly correlated due to different measurement noise



Data visualization

Suppose we want to explore the `USarrest` dataset[1], which contains crime statistics per 100.000 residents in 50 USA states. The variables are:

1. Murder arrests (per 100.000)
2. Assault arrests (per 100.000)
3. Percent urban population
4. Rape arrests (per 100.000)

State	Murder	Assault	Urban pop	Rape
Alabama	13.2	236	58	21.2
Alaska	10	260	48	44.5
Arizona	8.1	294	80	31
⋮	⋮	⋮	⋮	⋮

How to visualize a 4-dimensional dataset?

[1] https://forge.scilab.org/index.php/p/rdataset/source/tree/master/csv/datasets/USArrests.csv

Outline

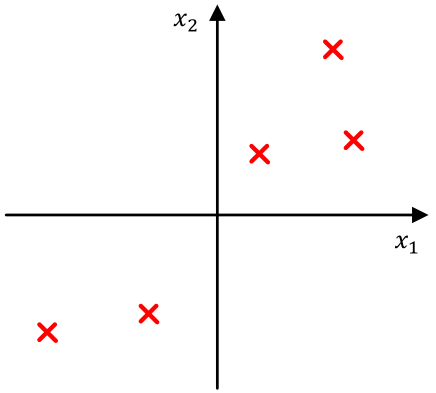
1. Introduction to Principal Component Analysis (PCA)
2. **PCA problem formulation**
3. Choice of the number of components
4. PCA-based process monitoring indicators

Problem formulation

Onto which **direction** it is better to **project** the data?

Pick the direction which **minimizes the projection error**.

This direction is also that on which the data **vary the most**

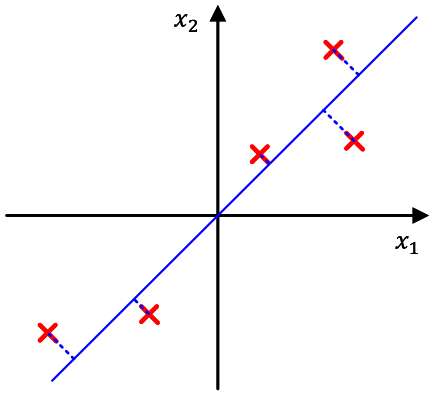


Problem formulation

Onto which **direction** it is better to **project** the data?

Pick the direction which **minimizes the projection error**.

This direction is also that on which the data **vary the most**

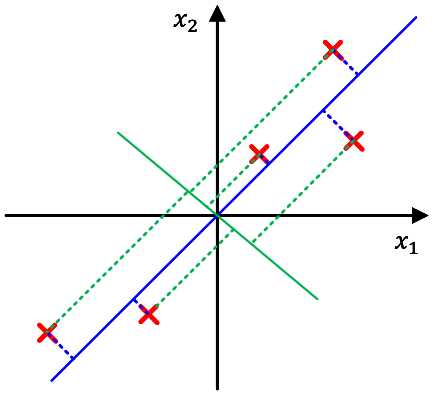


Problem formulation

Onto which **direction** it is better to **project** the data?

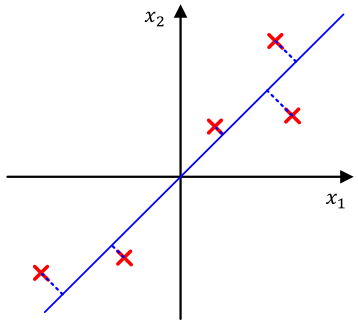
Pick the direction which **minimizes the projection error**.

This direction is also that on which the data **vary the most**

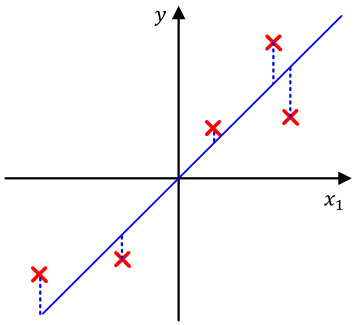


PCA is NOT Linear regression

PCA



Linear regression



PCA algorithm

Inputs

- The training set $\mathcal{D} = \{\boldsymbol{\varphi}(1), \boldsymbol{\varphi}(2), \dots, \boldsymbol{\varphi}(N)\}$, $\boldsymbol{\varphi} \in \mathbb{R}^d$ (do not consider dummy variables $\varphi_0 = 1$)
- q: the number of dimensions to retain (can be $q = d$)

Data pre-processing

- Remove the feature (column) mean for each column of the data matrix $X \in \mathbb{R}^{N \times d}$
- Standardize each feature element for the respective feature standard deviation
- Each feature now has mean 0 and standard deviation 1
- Define the normalized data matrix as $\tilde{X} \in \mathbb{R}^{N \times d}$

PCA algorithm

Perform a **Singular Value Decomposition** (SVD) of the matrix $\tilde{X} \in \mathbb{R}^{N \times d}$

$$\underset{N \times d}{\tilde{X}} = \underset{N \times N}{U} \cdot \underset{N \times d}{\Sigma} \cdot \underset{d \times d}{V^{\top}}$$

- The columns $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_d$ of V are called **principal component loadings** and they are the **eigenvectors** of the covariance matrix of the data $\frac{1}{N}\tilde{X}^{\top}\tilde{X}$
- Select the first $q \leq d$ columns of V, obtaining the **reduced** matrix $V_q \in \mathbb{R}^{d \times q}$
- Compute the projected data $Z = \tilde{X}V_q \in \mathbb{R}^{N \times q}$ **(principal component scores)**

Outline

Choice of the number of components

The singular values $\sigma_1, \sigma_2, \ldots$ in the matrix Σ can be used to compute the data **variance explained** by each principal component

The percentage of variance explained by the j-th component is:

$$e_j = \frac{\sigma_j^2}{\sum_{i=1}^{r} \sigma_i^2} \cdot 100$$

- r is the rank of $\tilde{X}$

Choices:

- Retain a number of components that explain a **determined level** of variance in the data
- Retain a **fixed number** of components

Example: USarrest data

The first 2 principal component loadings for the USarrest dataset are the first 2 columns of the matrix $V \in \mathbb{R}^{4 \times 4}$. Thus, $V_q \in \mathbb{R}^{4 \times 2}$

State	Murder	Assault	Urban pop	Rape
Alabama	13.2	236	58	21.2
Alaska	10	260	48	44.5
Arizona	8.1	294	80	31
⋮	⋮	⋮	⋮	⋮

PCA

Feature	PC 1	PC 2
Murder	−0.5359	0.4182
Assault	−0.5832	0.1880
UrbanPop	−0.2782	−0.8728
Rape	−0.5432	−0.1673

- It is possible to plot both the **projected data** and the **new dimensions**
- For example, the direction of the **Murder feature** is the direction of the line from the origin to the point $[-0.5359,\ \ 0.4182]$

Biplot

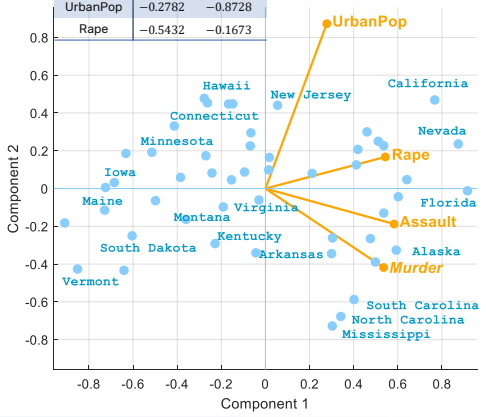
The 1st loading vector gives similar weight on **Assault**, **Murder**, and **Rape**, and less weight on **UrbanPop**

The 1st component roughly corresponds to a measure of **overall rates of serious crimes**

The 2nd loading vector places most of its weight on **UrbanPop** and much less weight on the other three features

The 2nd component roughly corresponds to the **level of urbanization** of the state

Feature	PC 1	PC 2
Murder	−0.5359	0.4182
Assault	−0.5832	0.1880
UrbanPop	−0.2782	−0.8728
Rape	−0.5432	−0.1673



Biplot

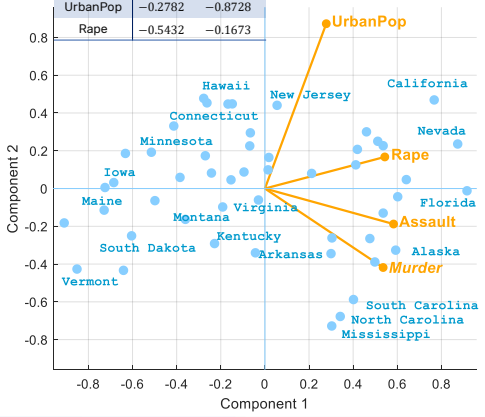
The **crime-related** variables (**Murder**, **Assault**, and **Rape**) are located close to each other, and the **UrbanPop** variable is far from the other three

⇩

The crime variables are correlated with each other (states with high murder rates tend to have high assault and rape rates)

The **UrbanPop** variable is less correlated with the other three

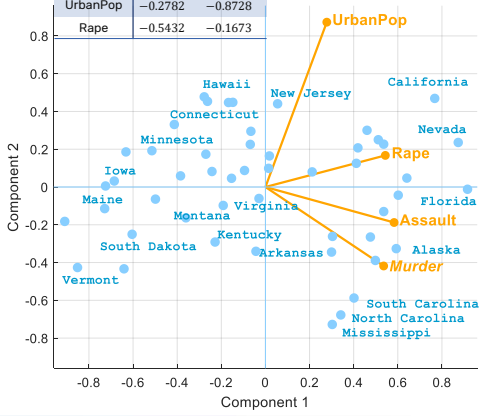
Feature	PC 1	PC 2
Murder	−0.5359	0.4182
Assault	−0.5832	0.1880
UrbanPop	−0.2782	−0.8728
Rape	−0.5432	−0.1673



Biplot

- States with **large positive scores on the 1st component**, such as California, Nevada and Florida, have **high crime rates**, while states like South Dakota, with negative scores on the first component, have low crime rates
- California also has a **high score on the 2nd component**, indicating a **high level of urbanization**, while the opposite is true for states like Mississippi
- States **close to zero** on both components, such as Virginia, have approximately **average levels** of both crime and urbanization

Feature	PC 1	PC 2
Murder	−0.5359	0.4182
Assault	−0.5832	0.1880
UrbanPop	−0.2782	−0.8728
Rape	−0.5432	−0.1673



Outline

1. Introduction to Principal Component Analysis (PCA)
2. PCA problem formulation
3. Choice of the number of components
4. **PCA-based process monitoring indicators**

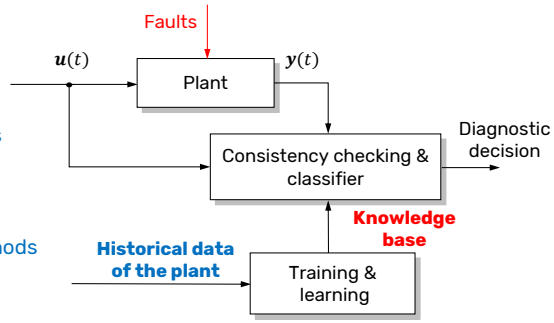
Knowledge-based fault diagnosis

Great amount of historical data available:

- There is **not a prior knowledge** about the faults, and it has to be **extracted from data**
- **Training vs testing** phases

Common approaches are:

- Machine learning supervised classifiers
- Unsupervised anomaly detection
- Transfer learning
- Statistical Process Control (SPC) methods (control charts, PCA indicators,…)



Knowledge-based fault diagnosis

For **large-scale processes**, such as chemical ones, the development of a plant model may be **too time consuming**. Knowledge-based (data-driven) methods offer an alternative

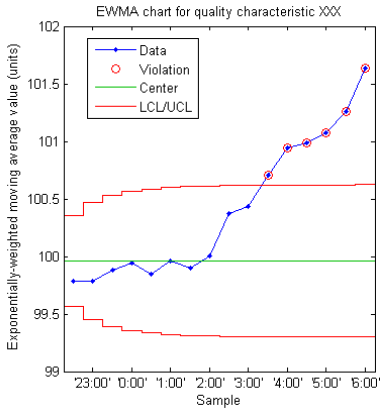
These methods are attractive when the available process **measurements** are **highly correlated**, so that they can be projected to a lower dimension

Statistical Process Monitoring (SPM) consists in a set of statistical-based techniques for evaluating anomalous behaviour in the data

Introduction to Statistical Process Monitoring (SPM)

The simplest Statistical Process Monitoring (SPM) approaches consist of univariate statistical **control charts:**

- Shewart chart
- CUSUM (Cumulative Sum)
- EWMA (Exponentially Weighted Moving Average)



Introduction to Statistical Process Monitoring (SPM)

Univariate control charts do not take into account the **correlation** between variables. So, **multivariate** methods are needed

PCA-based indicators are one of the most famous multivariate SPM approaches. The idea is to build a **data matrix** $X \in \mathbb{R}^{N \times d}$ with N **observations** of d process **measurements**

These indicators assume a **static** setting, i.e. they must consider data from the plant in **steady-state.** Model-based methods can deal with dynamic settings

Introduction to Statistical Process Monitoring (SPM)

In PCA, the **projected** data (scores) can be computed as $Z = \tilde{X} \cdot V_q \in \mathbb{R}^{N \times q}$

It is possibile to **reconstruct** the data, using only q components, as $\tilde{X}_r = Z \cdot V_q^\top \in \mathbb{R}^{N \times d}$

The reconstruced data matrix therefore reads as $\tilde{X}_r = \tilde{X} \cdot V_q \cdot V_q^\top \in \mathbb{R}^{N \times d}$

The data matrix $\tilde{X}$ can be decomposed as $\tilde{X} = \tilde{X}_r + E$. The matrix $E \in \mathbb{R}^{N \times d}$ is called **residual matrix**

The space $\mathcal{S}_v = \mathcal{C}(V_q)$ is called the **principal component subspace (PCS)**. Its orthogonal complement $\mathcal{S}_e = \mathcal{N}(V_q^\top)$ is called the **residual subspace (RS)**

T^2 statistic

The Hotelling's T^2 index measures variations in the **principal component subspace**

$$\underset{1\times 1}{T^2(\boldsymbol{\varphi})} \coloneqq \underset{1\times d}{\tilde{\boldsymbol{\varphi}}^\top} \underset{d\times q}{V_q} \cdot \underset{q\times q}{\Sigma_q^{-2}} \cdot V_q^\top \tilde{\boldsymbol{\varphi}} = \underset{1\times q}{\boldsymbol{z}^\top} \cdot \Sigma_q^{-2} \cdot \boldsymbol{z}$$

where Σ_q consists in the first q rows and q columns of the matrix Σ and $\tilde{\boldsymbol{\varphi}}$ is the standardized features vector

The T^2 statistic is a **scaled squared 2-norm** of an observation vector $\boldsymbol{\varphi}$ from its mean. The scaling on $\boldsymbol{\varphi}$ is in the direction of the q **eigenvectors** and is inversely proportional to the **standard deviation** along the eigenvectors (each σ_i^2 equals the eigenvalue λ_i of the covariance matrix $\tilde{X}^\top \tilde{X}$).

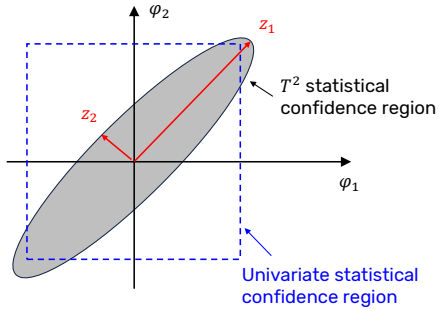
This allows a scalar threshold to characterize the data variability in the entire $\mathbb{R}^{q\times 1}$ space

T^2 statistic

Given a level of significance α, appropriate **thresholds** T_α^2 for the T^2 statistic can be determined **automatically**

Given an observation vector $\boldsymbol{\varphi}$, we have that:

1. The observation is **in-control** if $T^2(\boldsymbol{\varphi}) \leq T_\alpha^2$
2. The observation is **out-of-control** (there is a fault) if $T^2(\boldsymbol{\varphi}) > T_\alpha^2$



The confidence region is an **ellipsoid** that «streches» depending on the correlation bewteen the variables

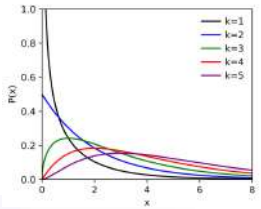
T^2 statistic

The T^2 statistic can be interpreted as measuring the **systematic variations** of the process, and a violation of the threshold indicates that these variations are out-of-control

A usual **assumption** is that the observations $\boldsymbol{\varphi}$ follow a multivariable Normal distribution

If the **mean** and the **covariance** matrix of the observations are **known** (or if not but N is high), the T^2 statistic follows a Chi-square distribution, and a threshold can be set as

$$T_\alpha^2 = \chi_\alpha^2(q)$$



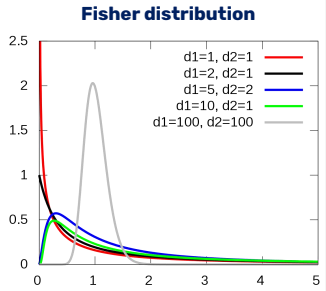
where $\chi_\alpha^2(q)$ indicates the $(1-\alpha)$ percentile of a Chi-square distribution with q degrees of freedom

T^2 statistic

If the mean and the covariance matrix are **not known but estimated**, the T^2 statistic follows a Fisher distribution, and a threshold can be set as

$$T_\alpha^2 = \frac{q(N-1)(N+1)}{N(N-q)} F_\alpha(q, N-q)$$

where $F_\alpha(q, N-q)$ indicates the $(1-\alpha)$ percentile of the Fisher distribution with q and $N-q$ degrees of freedom



SPE (Q) statistic

The Q statistic, also known as the **squared prediction error (SPE),** is a squared 2-norm measuring the deviation of the observations projected on the **residual subspace**

$$\underset{1\times 1}{Q(\boldsymbol{x})} \coloneqq \boldsymbol{e}^\top \boldsymbol{e} \qquad \underset{d\times 1}{\boldsymbol{e}} \coloneqq \tilde{\boldsymbol{\varphi}} - \tilde{\boldsymbol{\varphi}}_r = \left(I_d - \underset{d\times q}{V_q} \cdot \underset{q\times d}{V_q^\top}\right) \underset{d\times 1}{\tilde{\boldsymbol{\varphi}}}$$

The SPE statistic measures the random **variations** of the process, for example, that are associated with **measurement noise**

If the SPE statistic threshold are exceeded, it may indicate that the random measurements noise significantly changed

SPE (Q) statistic

The **threshold** Q_α for the SPE statistic can be computed as

$$Q_\alpha = g^{\mathrm{SPE}} \chi_\alpha^2 \left(h^{\mathrm{SPE}}\right)$$

$$g^{\mathrm{SPE}} = \frac{\theta_2}{\theta_1} \qquad h^{\mathrm{SPE}} = \frac{\theta_1^2}{\theta_2}$$

$$\theta_1 = \sum_{i=q+1}^{d} \sigma_i^2 \qquad \theta_2 = \sum_{i=q+1}^{d} \left(\sigma_i^2\right)^2$$

where $\chi_\alpha^2\left(h^{\mathrm{SPE}}\right)$ indicates the $(1-\alpha)$ percentile of a Chi-square distribution with h^{SPE} degrees of freedom

T^2 and SPE statistics

Since the **principal component subspace** typically contains **normal process variations** with large variance that represent signals, and the **residual subspace** contains mainly **noise**, the normal region defined by the control limit for T^2 is usually much larger than that of SPE

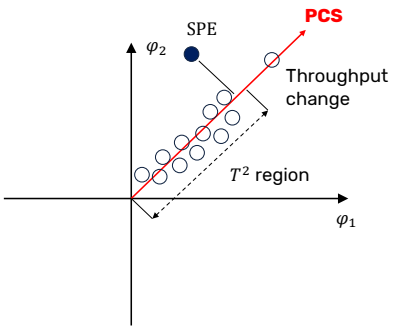
Thus, it usually takes a much **larger fault magnitude** to exceed the T^2 control limit

The normal region defined by the SPE control limit includes residual components that are mainly noise. Therefore, faults with **small to moderate magnitudes** can easily exceed the SPE control limit

T^2 and SPE statistics

Furthermore, if a sample exceeds the T^2 limit only, but does not violate the SPE limit, it does not break the correlation structure but simply shifts further away from the origin in the PCS

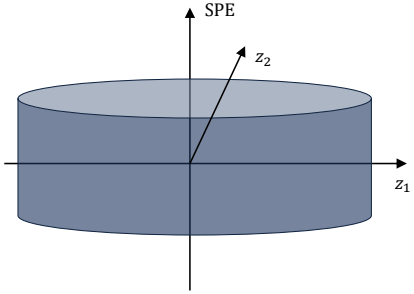
This case could be a fault, but it **could also be a change in the operating region** which is not necessarily a fault



T^2 and SPE statistics

The T^2 and Q statistics along with their appropriate thresholds detect **different types of faults**, and the advantages of both statistics can be utilized by employing the two measures together.

When these two statistics are utilized along with their respective thresholds, it produces a **cylindrical in-control region**



T^2 and SPE statistics

A monitoring algorithm based on the T^2 and Q statistics can be summarized as follows:

Phase 1: design or set-up

1. Using X, compute the mean $\hat{\mu}_j$ and the standard deviation $\hat{\sigma}_j$ of each feature j
2. Using $\hat{\mu}_j, \hat{\sigma}_j$ normalize training data X to zero mean and standard deviation one, $\bar{X}$
3. Compute $U\Sigma V^\top \leftarrow \mathrm{svd}(\bar{X})$. Select q and compute V_q, Σ_q
4. Select α and compute the thresholds T^2_α, Q_α

Phase 2: usage

1. Normalize a new data $\boldsymbol{\varphi}^*$ using $\hat{\mu}_j$ and $\hat{\sigma}_j$
2. Compute the statistics $T^2(\boldsymbol{\varphi}^*)$ and $Q(\boldsymbol{\varphi}^*)$ using V_q, Σ_q and Σ
3. Compare $T^2(\boldsymbol{\varphi}^*)$ with T_α and $Q(\boldsymbol{\varphi}^*)$ with Q_α to detect abnormalities in process data

Extensions and further considerations

The presented rationales are used for fault detection. Fault isolation can be performed by **contribution plots**, which compute the contribution of each variable to the fault decision

The PCA model can be extended to consider dynamical systems (**dynamic PCA**). If we suppose that $\boldsymbol{\varphi}(t) = [y_t \;\; u_t]^\top \equiv [y(t)\, u(t)]^\top$, we can define a matrix $X(h)$, where h defines the order of the system, as

$$X(h) = \begin{bmatrix} y_t & u_t & y_{t-1} & u_{t-1} & \cdots & y_{t-h} & u_{t-h} \\ y_{t-1} & u_{t-1} & y_{t-2} & u_{t-2} & & y_{t-h-1} & u_{t-h-1} \\ \vdots & \vdots & \vdots & \vdots & \ddots & & \vdots \\ y_{t+h-N} & u_{t+h-N} & y_{t+h-N-1} & u_{t+h-N-1} & \cdots & y_{t-N} & u_{t-N} \end{bmatrix}$$

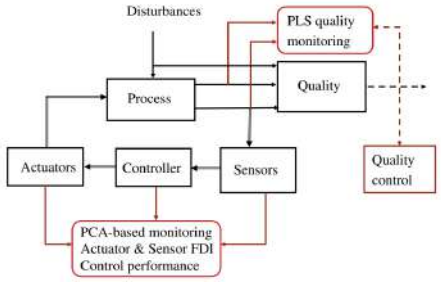
Data preprocessing

Prior to the construction of the data matrix X, the data should be **preprocessed** by:

- removing variables that are known to be **not important** for the monitoring purpose
- removing **outliers** from measurements

Extensions and further considerations

The method of **Partial Least Squares (PLS)** is a dimensionality reduction technique for **maximizing the covariance** between the predictor (independent) matrix X and the predicted (dependent) matrix Y for each component of the reduced space



A popular application of PLS is to select the matrix Y to contain only **product quality data** which can even include off-line measurement data, and the matrix X to contain all other **process variables**

UNIVERSITÀ DEGLI STUDI DI BERGAMO | Dipartimento di Ingegneria Gestionale, dell'Informazione e della Produzione