LECTURE NOTES OF

MODEL IDENTIFICATION AND DATA ANALYSIS PART 2

From Professor Sergio Savaresi's lectures

Authors and Contributors: EDOARDO MORASSUTTO, MARCO DONADONI, COSIMO RUSSO, FEDERICO CAZZOLA

Reviewed and updated for the 2022 edition of the course by
ANDREA BOSISIO

Politecnico di Milano
A.Y. 2021/2022

Lecture Notes of Model Identification and Data Analysis - Part 2

Revision of May 29, 2022

Please notify errors or changes through an email or a pull request.

Contact:
Andrea Bosisio - `andrea2.bosisio@mail.polimi.it`

Contents

Chapter 0

Introduction

0.1 Prerequisites

The *Model Identification and Data Analysis - Part 2*[1] course is a graduate level course of the MSc in Computer Science and Engineering held at Politecnico di Milano; hence, familiarity with basic concepts of computer science (algorithms and complexity), dynamical systems theory and a mathematical maturity in linear algebra and probability theory are prerequisites.

Furthermore, in order to understand the concepts of this course it is recommended to firstly follow the *Model Identification and Data Analysis - Part 1*[2] course.

0.2 General topics of MIDA course

- Collect digitally data from real systems
- Build Black-Box (BB) (or GB) models from data, with emphasis on
  - Dynamic systems
  - Control/automation-oriented applications
- Purpose of modelling (area of machine leasing focusing on "control")
  - Prediction
  - Software-sensing
  - Modelling for control design

0.2.1 Super summary of MIDA1

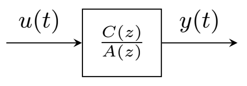
The focus is on *Time Series* (output-only systems) and *input/output* (I/O) systems.

Models used in MIDA1:

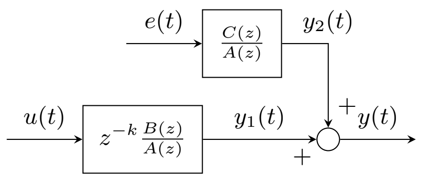
- Auto Regressive Moving Average (ARMA) models for time series

[1]For more information about the MIDA course see here.
[2]MIDA1 lecture notes available here.

- Auto Regressive Moving Average with Exogenous Input (ARMAX) models for I/O systems



ARMA model



ARMAX model

We've used a *Black-Box model identification* of the model, indicated as $\mathcal{M}(\vartheta)$ where ϑ is the parameter vector, which contains the coefficients of $A(z)$, $B(z)$, $C(z)$.

In particular we've seen the **PEM** method, a *parametric approach* based on the minimization of the *performance index* defined as:

Definition 0.1 — PEM cost function. $J_N(\vartheta) = \frac{1}{N}\sum_{t=1}^{N}\left(y(t) - \widehat{y}(t|t-1,\vartheta)\right)^2$

which is the variance of the *prediction error* made by the model. The optimal ϑ is $\widehat{\vartheta}_N = \arg\min_{\vartheta} J_N(\vartheta)$

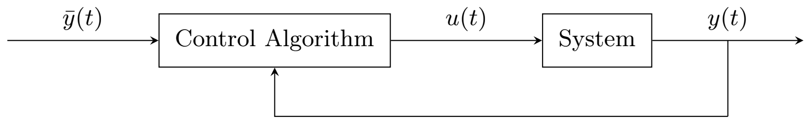
0.2.2 MIDA 2

The focus is on *Dynamical System* with a special eye on control (or feedback) application (more close to real applications than time series). We can divide the course in three main blocks: *Advanced Model Identification* methods, *Software-Sensing (SW)* and *Minimum Variance Control*.
In particular we will see:

- Non-parametric (direct/constructive) BB identification of I/O systems using State-Space models
- Parametric identification for BB I/O systems, with a frequency-domain approach
- Kalman-filter for SW-sensing using feedback on White-Box (WB) models
- BB methods for SW-sensing without feedback
- GB system identification using Kalman-filter and using *simulation-error methods* (SEM)
- Minimum-Variance Control (MVC), design of optimal feedback controllers using the theory background of the MIDA course
- Recursive (online) implementation of algorithms for system identification

0.3 Motivation example for the course: ABS (Anti-Lock Braking System)

ABS is an example of a control system since it follows this general scheme:

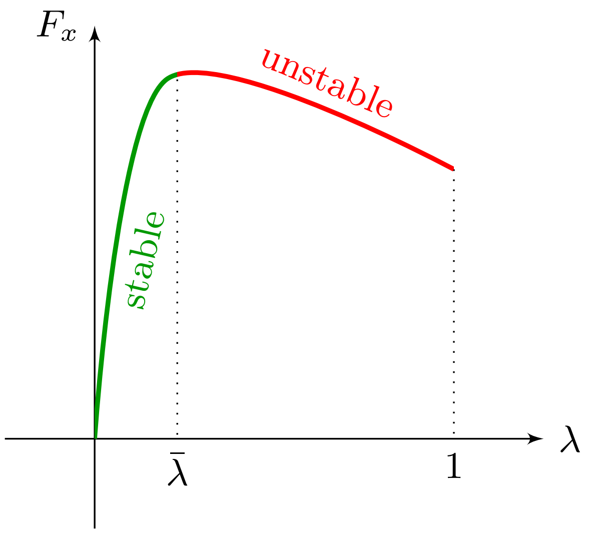


Control System Scheme

where $\bar{y}(t)$ is the reference value of $y(t)$.

We define the *slip* of the wheel as $\lambda(t) = \frac{v(t)-\omega(t)r}{v(t)}$, where $v(t)$ is the horizontal velocity of the car, $\omega(t)$ is the angular velocity of the wheel and r is the radius of the wheel.

During a brake $0 \leq \lambda(t) \leq 1$ (from free rolling wheel (i.e. $v(t) = \omega(t)r$) to locked wheel (i.e. $\omega(t) = 0$)).
The curve of F_x, the braking force, is as showed below:



Relation between λ and the braking force.

In the case of ABS, $u(t) = x(t)$ which is the voltage of the electric braking motor (control variable), $y(t) = \lambda(t)$ (controlled variable) and $\bar{y}(t) = \bar{\lambda}(t)$, which is the maximum braking point that we want to reach during an emergency brake.

The problem can be divided into subproblems:

- Model of the system
- SW-estimation of λ since v is **not** directly measurable, so λ cannot be computed
- Design of the ABS control algorithm

Because of that measurement problem we have to build a Black-Box model from data.

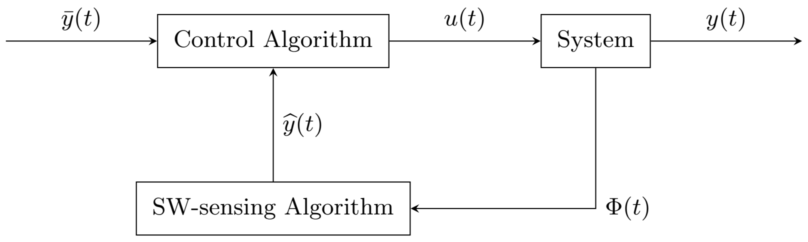
Why Black-Box modelling? The control variable x (the voltage to the actuator) controls a complex systems from the actuator to λ. The system can be seen as a chain of components:

- Current dynamics and electric motor
- Position dynamics of the actuator
- Dynamics of the hydraulic circuit of the braking system
- Tire dynamics
- Wheel rotational dynamics
- Vehicle full dynamics

It's simply deducible that such system is really difficult to model.

> **Recall — White-Box (WB) vs. Black-Box (BB).**
>
> - WB modelling: write the physical equations from *first principles.*
> - BB modelling: experiment $\rightarrow$ collect data $\rightarrow$ build model. Using only I/O measured data we can *learn* a mathematical model of the I/O behavior of the system.

In order to estimate the variable $v(t)$ (and so also $\lambda(t)$) we use an SW-sensing algorithm which takes as inputs other measurable variables (e.g. angular velocities of all the wheels). Thus, the scheme of a general control system become something like this:

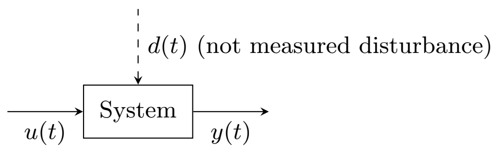


Control System Scheme with SW-sensing

where $\Phi(t)$ are the available (measurable) variables of the system and $\widehat{y}(t)$ is the estimation of the SW-sensing algorithm of $y(t)$

Chapter 1

Black-Box non-parametric identification of I/O systems using State-Space models in time domain



Recall — General path of a Parametric Identification Methods.

1. Collect data: inputs: $\{u(1), u(2), \ldots, u(N)\}$, outputs: $\{y(1), y(2), \ldots, y(N)\}$
2. Select **a-priori** a class/family of **parametric models**: $\mathcal{M}(\vartheta)$
3. Select **a-priori** a performance index $J(\vartheta)$ (which gives an order to the quality of the models)
4. Optimization step (minimize $J(\vartheta)$ w.r.t ϑ): $\widehat{\vartheta}_N = \arg\min_{\vartheta} J(\vartheta) \rightarrow$ optimal model $\mathcal{M}(\widehat{\vartheta}_N)$ characterized by the **optimal parameters** $\widehat{\vartheta}_N$

Note: $J(\vartheta) : \mathbb{R}^{n_\vartheta} \rightarrow \mathbb{R}^+$ (where n_ϑ is the order of the model).

Note: $\mathcal{M}(\vartheta_1)$ can be sorted, that is $\mathcal{M}(\vartheta)$ is better than $\mathcal{M}(\vartheta_2)$ if $J(\vartheta_1) < J(\vartheta_2)$.

In this chapter we are presenting a totally different system identification approach, the **non-parametric** one, which means:

- No a-priori model-class selection
- No performance index definition
- No optimization task

Before entering in the system identification algorithm we need to recall the three main mathematical representation of **Discrete-time Dynamic Linear Systems** which are characterized by their internal variables called *states* and indicated with $x(t)$.

1.1 Representations

1.1.1 Representation #1: State-Space (SS)

$$\mathcal{S}: \begin{cases} x(t+1) = Fx(t) + Gu(t) & \text{state equations} \\ y(t+1) = Hx(t) + Du(t) & \text{output equations} \end{cases}$$

where F, G, H and D are matrices defined a follows:

$$F = \begin{bmatrix} \\ n \times n \\ \text{state matrix} \\ \\ \end{bmatrix} \qquad G = \begin{bmatrix} \\ n \times 1 \\ \text{input} \\ \text{matrix} \\ \\ \end{bmatrix}$$

$$H = \begin{bmatrix} 1 \times n & \text{output matrix} \end{bmatrix} \qquad D = \begin{bmatrix} 1 \times 1 & \text{i/o matrix} \end{bmatrix}$$

In this case we have 1 input and 1 output (i.e. *SISO* system), but those *difference equations* can be extended for multiple inputs and outputs systems. Usually $D = 0$ since we can say that the majority of real systems have this property.

Definition 1.1 — Strictly-proper system. A system is called *strictly-proper* when the output of the system doesn't directly depend on the input (i.e. $D = 0$).

Observation. n is the *order* of the system.

Example — SISO system of order $n = 2$.

$$\mathcal{S}: \begin{cases} x_1(t+1) = \frac{1}{2}x_1(t) + 2u(t) \\ x_2(t+1) = x_1(t) + 2x_2(t) + u(t) \\ y(t) = \frac{1}{4}x_1(t) + \frac{1}{2}x_2(t) \end{cases}$$

In this case $n = 2$, $x(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix}$, one input $u(t)$ and one output $y(t)$.

$$F = \begin{bmatrix} \frac{1}{2} & 0 \\ 1 & 2 \end{bmatrix} \qquad G = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$
$$H = \begin{bmatrix} \frac{1}{4} & \frac{1}{2} \end{bmatrix} \qquad D = 0$$

Remark — State-Space representation is not unique. $F_1 = TFT^{-1}$, $G_1 = TG$, $H_1 = HT^{-1}$, $D_1 = D$ for any invertible $(n \times n)$ matrix T. The system $\{F, G, H, D\}$ is equivalent to $\{F_1, G_1, H_1, D_1\}$.

1.1.2 Representation #2: Transfer Function (TF)

$$W(z) = \frac{B(z)}{A(z)} z^{-k} = \frac{b_0 + b_1 z_{-1} + b_2 z^{-2} + \ldots + b_p z^{-p}}{a_0 + a_1 z^{-1} + a_2 z^{-2} + \ldots + a_n z^{-n}} z^{-k}$$

$$\mathcal{S} : y(t) = W(z)u(t)$$

The Transfer Function (TF) $W(z)$ is a rational function of z: it's a *digital filter*.

Recall — z operator. The linear z operator is such that $z^{-1}[x(t)] = x(t-1)$ and $z[x(t)] = x(t+1)$.
From now on the "$[\cdot]$" will be **omitted** for simplicity.

Example.

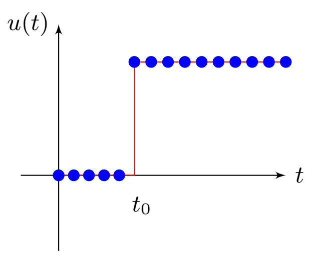
$$\mathcal{S} : \quad y(t) = \underbrace{\left[\frac{1+\frac{1}{2}z^{-1}}{2+\frac{1}{3}z^{-1}+\frac{1}{4}z^{-2}} z^{-1}\right]}_{W(z)} u(t)$$

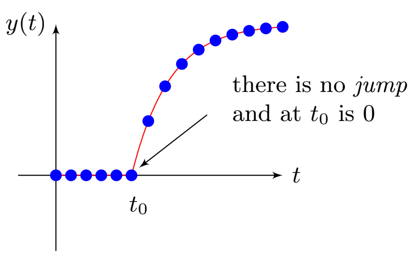
$$2y(t) + \frac{1}{3}y(t-1) + \frac{1}{4}y(t-2) = u(t-1) + \frac{1}{2}u(t-2)$$

$$y(t) = \underbrace{-\frac{1}{6}y(t-1) - \frac{1}{8}y(t-2)}_{\text{recursive part}} + \underbrace{\frac{1}{2}u(t-1) + \frac{1}{4}u(t-2)}_{\text{past inputs}}$$

Remark — IRR and FIR filters.
$W(z) = \dfrac{z^{-1}}{1+\frac{1}{3}z^{-1}}$ is an IIR (*Infinite Impulse Response*) filter since it has the recursive part because of the presence of *poles* in $W(z)$.
$W(z) = z^{-1} + \dfrac{1}{2}z^{-2} + \dfrac{1}{4}z^{-3}$ is a FIR (*Finite Impulse Response*) filter since it depends only on a *finite* sequence of past inputs.

Remark — Strictly proper systems. Notice that for strictly proper systems the delay of $W(z)$ is $k \geq 1$, or, equivalently, the order of the numerator of $W(z)$ is strictly smaller than the order of the numerator of $W(z)$.



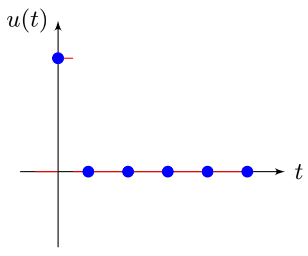


1.1.3 Representation #3: Convolution of the input with the Impulse Response (IR)

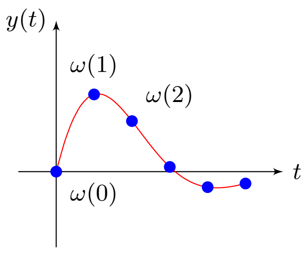
The third way to represent a system is through the *convolution* of the input with the *Impulse Response (IR).*

Definition 1.2 — Impulse Response. In the discrete time domain, the Impulse Response of a filter $W(z)$ is $y(t) = W(z)u(t)$ where the input $\omega(t)$ is an impulse (i.e. $u(t) = 0$ everywhere except from $t = 0$ where $u(t = 0) = 0$).

$$\omega(t) = \{\omega(0), \omega(1), \omega(2), \cdots\}$$



Impulse in input



Impulse Response (IR) in output

Note if the system is strictly proper then $\omega(0) = 0$.

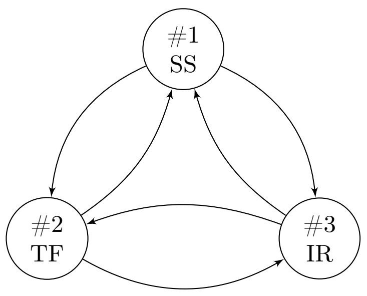
It can be proven that the input-output relationship from $u(t)$ to $y(t)$ can be written as

$$y(t) = \omega(0)u(t) + \omega(1)u(t-1) + \omega(2)u(t-2) + \cdots = \sum_{k=0}^{\infty} \omega(k)u(t-k)$$

From this, it is clear that $y(t)$ is the *convolution* of the IR with the input signal.

1.2 Transformations between representations

It is possible translate each representation into another one. Therefore, there are six possible transformations between the three representations.



Transformations between representations

1.2.1 State-Space to Transfer Function

Consider a strictly proper system with the following State-Space (SS) representation:

$$\mathcal{S}: \begin{cases} x(t+1) = Fx(t) + Gu(t) \\ y(t) = Hx(t) + \cancelto{0}{Du(t)} \end{cases} \Rightarrow \begin{cases} x(t+1) = Fx(t) + Gu(t) \\ y(t) = Hx(t) \end{cases}$$

From the system we get

$$zx(t) = Fx(t) + Gu(t) \Rightarrow x(t) = (zI - F)^{-1}Gu(t)$$

$$\Rightarrow y(t) = H(zI - F)^{-1}G \cdot u(t)$$

Thus, the Transfer Function is

$$\text{SS}\rightarrow\text{TF:} \qquad W(z) = H(zI - F)^{-1}G \tag{1.1}$$

Example. Consider the following SISO system of order $n = 2$:

$$F = \begin{bmatrix} 1 & 0 \\ \frac{1}{2} & 2 \end{bmatrix} \qquad G = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \qquad H = \begin{bmatrix} 1 & 0 \end{bmatrix} \qquad D = 0$$

Using the transformation 1.1:

$$\begin{aligned} W(z) &= \begin{bmatrix} 1 & 0 \end{bmatrix} \left(\begin{bmatrix} z & 0 \\ 0 & z \end{bmatrix} - \begin{bmatrix} 1 & 0 \\ \frac{1}{2} & 2 \end{bmatrix} \right)^{-1} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} z-1 & 0 \\ -\frac{1}{2} & z-2 \end{bmatrix}^{-1} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} 1 & 0 \end{bmatrix} \frac{1}{(z-1)(z-2)} \begin{bmatrix} z-2 & 0 \\ \frac{1}{2} & z-1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \frac{1}{(z-1)(z-2)} \begin{bmatrix} z-2 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \\ &= \frac{\cancel{z-2}}{(z-1)\cancel{(z-2)}} = \frac{1}{z-1} = \frac{1}{1-z^{-1}} z^{-1} \end{aligned}$$

Notice that, due to **cancellation of singularities**, in this case we only have one pole, but the system is of order two; this comes from the fact that part of the system is *non observable.*
Alternatively, it could be noted that $\{F, G, H, D\}$ corresponds to the following system:

$$\mathcal{S}: \begin{cases} x_1(t+1) = x_1(t) + u(t) \\ x_2(t+1) = \frac{1}{2}x_1(t) + 2x_2(t) + u(t) \\ y(t) = x_1(t) \end{cases} \Rightarrow \begin{cases} zx_1(t) = x_1(t) + u(t) \\ zx_2(t) = \frac{1}{2}x_1(t) + 2x_2(t) + u(t) \\ y(t) = x_1(t) \end{cases}$$

From the first equation we have that $x_1(t) = \frac{1}{z-1}u(t)$ and substituting it to the third one we obtain the same result: $y(t) = x_1(t) = \frac{1}{z-1}u(t) \Rightarrow W(z) = \frac{1}{1-z^{-1}}z^{-1}$

1.2.2 Transfer Function to State-Space

This conversion is not very used in practice and it is called the *realization* of a Transfer Function into a State-Space system.

Issue: the State-Space representation is not unique! Thus from a single Transfer Function we can get infinite different equivalent State-Space models.

1.2.2.1 Control realization

We assume that the system is **strictly proper** and that the denominator is **monic** (i.e. $a_0 = 1$).

$$W(z) = \frac{b_0 z^{n-1} + b_1 z^{n-2} + \cdots + b_{n-1}}{z^n + a_1 z^{n-1} + a_2 z^{n-2} + \cdots + a_n}$$

The formula for the control realization of $W(z)$ is

$$\text{TF}\to\text{SS:}\quad F = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ 0 & 0 & \cdots & 0 & 1 \\ -a_n & -a_{n-1} & & \cdots & -a_1 \end{bmatrix} \quad G = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix} \quad H = \begin{bmatrix} b_{n-1} & b_{n-2} & \cdots & b_0 \end{bmatrix} \quad D = 0 \tag{1.2}$$

Example. Consider the following Transfer Function:

$$W(z) = \frac{2z^2 + \frac{1}{2}z + \frac{1}{4}}{z^3 + \frac{1}{4}z^2 + \frac{1}{3}z + \frac{1}{5}}$$

Applying the transformation 1.2, the control realization is:

$$F = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -\frac{1}{5} & -\frac{1}{3} & -\frac{1}{4} \end{bmatrix} \qquad G = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \qquad H = \begin{bmatrix} \frac{1}{4} & \frac{1}{2} & 2 \end{bmatrix} \qquad D = 0$$

1.2.3 Transfer Function to Impulse Response

To get the IR from a Transfer Function $W(z)$ is sufficient to make the ∞-long division between the numerator and denominator of $W(z)$.

$$\text{TF}\to\text{IR:}\qquad \omega(t) = \begin{cases} e_t & \text{if } t \geq 1 \\ 0 & \text{if } t = 0 \text{ (and } W(z) \text{ is strictly-proper)} \end{cases} \tag{1.3}$$

where e_t are the coefficients of

$$E(z) = e_1 z^{-1} + e_2 z^{-2} + \cdots = \sum_{t=1}^{\infty} e_t z^{-t}$$

which is the *remainder* of the ∞-long division.

Example. Consider the following Transfer Function:

$$W(z) = \frac{1}{z - \frac{1}{2}} = \frac{z^{-1}}{1 - \frac{1}{2}z^{-1}} \overset{\infty\text{-long div.}}{=} 0z^{-0} + 1z^{-1} + \frac{1}{2}z^{-2} + \frac{1}{4}z^{-3} + \cdots$$

Applying the transformation 1.3 the IR is $\omega(0) = 0$, $\omega(1) = 1$, $\omega(2) = \frac{1}{2}$, $\omega(3) = \frac{1}{4}$, ... Thus $\omega(0) = 0$ and $\omega(t) = \frac{1}{2^{t-1}} \quad \forall t \geq 1$

In this case there is also a quicker way

$$y(t) = \frac{z^{-1}}{1 - \frac{1}{2}z^{-1}}u(t) = \left(z^{-1}\frac{1}{1 - \frac{1}{2}z^{-1}}\right)u(t)$$

Remembering that for *geometric series* we have

$$\sum_{k=0}^{\infty} a^k = \frac{1}{1-a} \text{ if } |a| < 1$$

We can rewrite $y(t)$ as follows

$$y(t) = \left(z^{-1}\sum_{k=0}^{\infty}\left(\frac{1}{2}z^{-1}\right)^k\right)u(t) = \left(0 + 1z^{-1} + \frac{1}{2}z^{-2} + \frac{1}{4}z^{-3} + \cdots\right)u(t)$$

1.2.4 Impulse Response to Transfer Function

Definition 1.3. Given a discrete-time signal $s(t)$ such that $\forall t < 0 : s(t) = 0$, its *Z-Transform* is defined as

$$\mathscr{Z}(s(t)) = \sum_{t=0}^{\infty} s(t)z^{-t}$$

Given this, it can be proven that

$$\text{TF}\to\text{IR:} \qquad W(z) = \mathscr{Z}(\omega(t)) = \sum_{t=0}^{\infty} \omega(t)z^{-t} \tag{1.4}$$

This means that the **Transfer Function of a system** is the $\mathscr{Z}$-transform of a special signal, $\omega(t)$, that is the Impulse Response of the system.

Remark. This formula cannot be used in practice to transform an IR representation to a TF representation. This is because we need infinite points of the Impulse Response, and it must be available *noise-free*. Thus, this transformation is only theoretical.

1.2.5 State-Space to Impulse Response

Consider the following State-Space model, with initial conditions $x(0) = 0$ and $y(0) = 0$

$$\mathcal{S} : \begin{cases} x(t+1) = Fx(t) + Gu(t) \\ y(t) = Hx(t) \end{cases}$$

"Running the simulation of the system", we have that

$$\begin{aligned}
&x(1) = \cancelto{0}{Fx(0)} + Gu(0) = Gu(0) \\
&y(1) = Hx(1) = HGu(0) \\
&\qquad\Downarrow \\
&x(2) = Fx(1) + Gu(1) = FGu(0) + Gu(1) \\
&y(2) = Hx(2) = HFGu(0) + HGu(1)
\end{aligned}$$

$$\Downarrow$$
$$x(3) = Fx(2) + Gu(2) = F^2Gu(0) + FGu(1) + Gu(2)$$
$$y(3) = Hx(3) = HF^2Gu(0) + HFGu(1) + HGu(2)$$
$$\vdots$$

This can be generalized to

$$y(t) = 0u(t) + HGu(t-1) + HFGu(t-2) + HF^2Gu(t-3) + \cdots$$

Recalling that

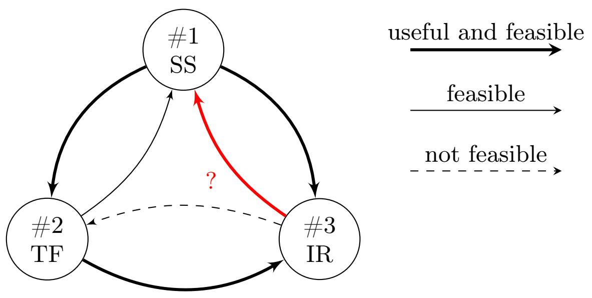
$$y(t) = \omega(0)u(t) + \omega(1)u(t-1) + \omega(2)u(t-2) + \omega(3)u(t-3) + \cdots$$

we deduce that $\omega(0) = 0$, $\omega(1) = HG$, $\omega(2) = HFG$, $\omega(3) = HF^2G$, ...
Thus, the Impulse Response is

$$\text{SS}\rightarrow\text{IR:} \qquad \omega(t) = \begin{cases} 0 \text{ if } t = 0 \\ HF^{t-1}G \text{ if } t > 0 \end{cases} \tag{1.5}$$

1.2.6 Summary of transformations

Notice that the IR representation is very easy to obtain experimentally, since we only need to measure the system response to the impulse signal. However, given the IR representation, it is difficult to get to the other representations, since the transformation from IR to TF is only theoretical. Moving from the IR to the SS representation is the key task of the *Subspace-based State-Space System Identification*, also known as *4SID method*.



Transformations between representations in practice

Before moving to the topic of *4SID method*, a recall of the *observability* and *controllability* properties for a linear dynamic system is needed.

1.3 Fundamental concepts of Observability and Controllability

$$\mathcal{S}: \begin{cases} x(t+1) = Fx(t) + Gu(t) \\ y(t) = Hx(t) \end{cases}$$

Definition 1.4 — Fully Observable. The system is fully observable (from the output) if and only if the **observability matrix** is full rank:

$$O = \begin{bmatrix} H \\ HF \\ \vdots \\ HF^{n-1} \end{bmatrix} \qquad \text{rank}(O) = n$$

where n is the order of the system.

Observation — Observability. Observability is a property of the system: by observing the output $y(t)$ it's possible to observe the state $x(t)$.
Note: It refers only to state and output (i.e. F and H).

Definition 1.5 — Fully Controllable. The system is fully controllable (from the input) if and only if the **controllability matrix** is full rank:

$$R = \begin{bmatrix} G & FG & \cdots & F^{n-1}G \end{bmatrix} \qquad \text{rank}(R) = n$$

where, again, n is the order of the system.
R is also called *reachability* matrix.

Observation — Controllability. Controllability is a property of the system: by driving (or move) the input $u(t)$ it's possible to control the state $x(t)$.
Note: It refers only to input and state (i.e. F and G).
Note: Controllability and Reachability are synonyms.

Example — SISO system of order $n = 2$.

$$\mathcal{S} : \begin{cases} x_1(t+1) = \frac{1}{2}x_1(t) + u(t) \\ x_2(t+1) = \frac{1}{3}x_2(t) \\ y(t) = \frac{1}{4}x_1(t) \end{cases} \qquad F = \begin{bmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{3} \end{bmatrix} \qquad H = \begin{bmatrix} \frac{1}{4} & 0 \end{bmatrix} \qquad G = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$O = \begin{bmatrix} H \\ HF \end{bmatrix} = \begin{bmatrix} \frac{1}{4} & 0 \\ \frac{1}{8} & 0 \end{bmatrix} \qquad \text{rank}(O) = 1 < n = 2 \implies \text{not fully observable}$$

$$R = \begin{bmatrix} G & FG \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 1 & \frac{1}{3} \end{bmatrix} \qquad \text{rank}(R) = 1 < n = 2 \implies \text{not fully controllable}$$

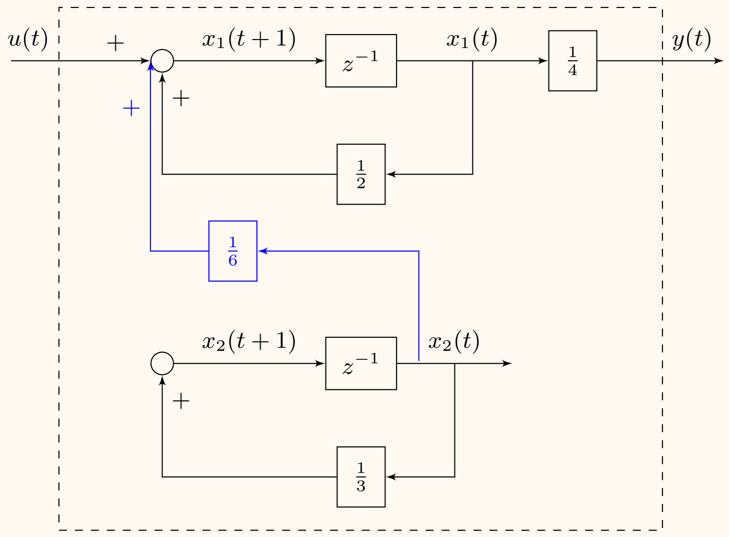
If $\mathcal{S}$ become

$$\mathcal{S}' : \begin{cases} x_1(t+1) = \frac{1}{2}x_1(t) + u(t) + \frac{1}{6}x_2(t) \\ x_2(t+1) = \frac{1}{3}x_2(t) \\ y(t) = \frac{1}{4}x_1(t) \end{cases} \qquad \text{only } F \text{ change and become} \quad F = \begin{bmatrix} \frac{1}{2} & \frac{1}{6} \\ 0 & \frac{1}{3} \end{bmatrix}$$

And

$$O = \begin{bmatrix} H \\ HF \end{bmatrix} = \begin{bmatrix} \frac{1}{4} & 0 \\ \frac{1}{8} & \frac{1}{24} \end{bmatrix} \qquad \text{rank}(O) = 2 = n \quad \Longrightarrow \quad \text{fully observable}$$

Observability and Controllability can also be checked graphically by building the block scheme



From this block scheme it can be seen that x_1 is *directly* observable and reachable; x_2 become only *indirectly* observable through $x_1(t)$ with the introduction of the blue block that correspond to the term $\frac{1}{6}x_2(t)$ in $\mathcal{S}'$.

Example — SISO system of order $n = 2$.

$$\mathcal{S}: \begin{cases} x_1(t+1) = \frac{1}{2}x_1(t) \\ x_2(t+1) = \frac{1}{3}x_2(t) + u(t) \\ y(t) = \frac{1}{4}x_1(t) \end{cases} \qquad F = \begin{bmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{3} \end{bmatrix} \qquad H = \begin{bmatrix} \frac{1}{4} & 0 \end{bmatrix} \qquad G = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$O = \begin{bmatrix} H \\ HF \end{bmatrix} = \begin{bmatrix} \frac{1}{4} & 0 \\ \frac{1}{8} & 0 \end{bmatrix} \qquad \text{rank}(O) = 1 < n = 2 \quad \Longrightarrow \quad \text{not fully observable}$$

$$R = \begin{bmatrix} G & FG \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 1 & \frac{1}{3} \end{bmatrix} \qquad \text{rank}(R) = 1 < n = 2 \quad \Longrightarrow \quad \text{not fully controllable}$$

If $\mathcal{S}$ become

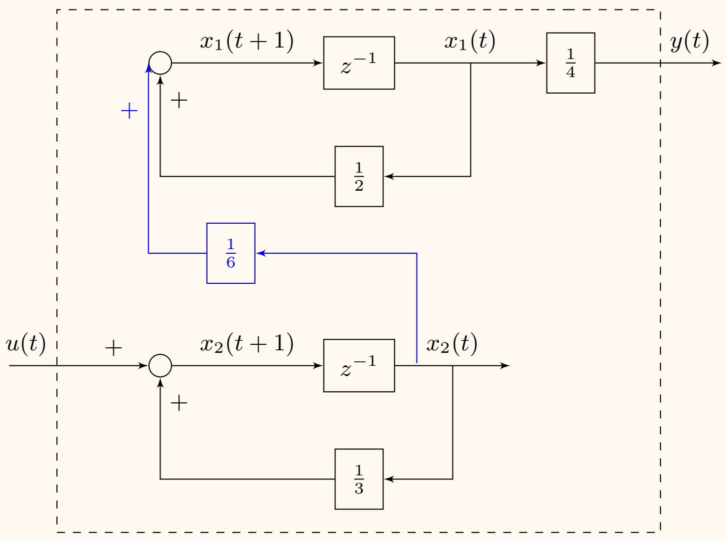
$$\mathcal{S}': \begin{cases} x_1(t+1) = \frac{1}{2}x_1(t) + \frac{1}{6}x_2(t) \\ x_2(t+1) = \frac{1}{3}x_2(t) + u(t) \\ y(t) = \frac{1}{4}x_1(t) \end{cases} \qquad \text{only } F \text{ change and become} \quad F = \begin{bmatrix} \frac{1}{2} & \frac{1}{6} \\ 0 & \frac{1}{3} \end{bmatrix}$$

And

$$O = \begin{bmatrix} H \\ HF \end{bmatrix} = \begin{bmatrix} \frac{1}{4} & 0 \\ \frac{1}{8} & \frac{1}{24} \end{bmatrix} \qquad \text{rank}(O) = 2 = n = 2 \quad \Longrightarrow \quad \text{fully observable}$$

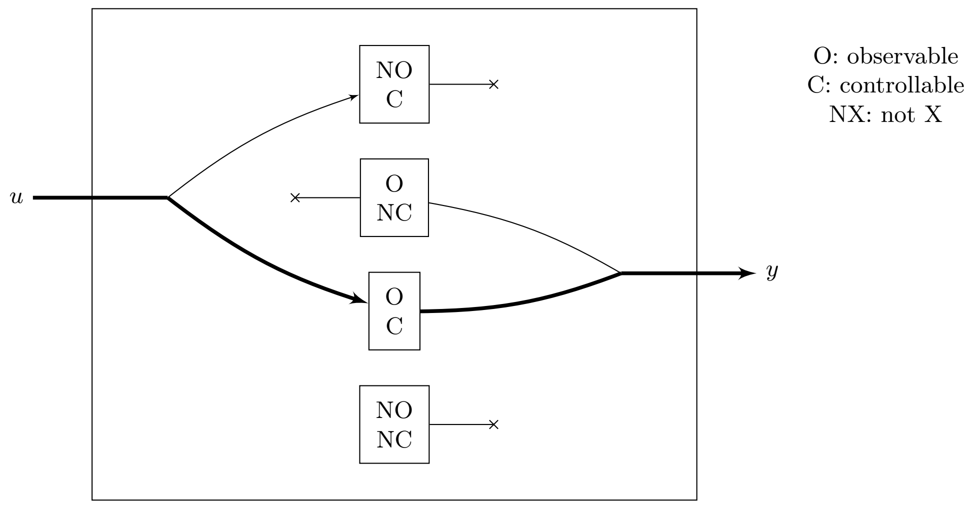
$$R = \begin{bmatrix} G & FG \end{bmatrix} = \begin{bmatrix} 0 & \frac{1}{6} \\ 1 & \frac{1}{3} \end{bmatrix} \qquad \text{rank}(R) = 2 = n = 2 \quad \Longrightarrow \quad \text{fully controllable}$$

Again, this properties can be also check using the block scheme

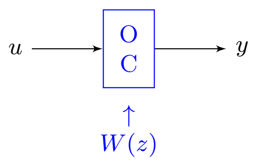


It can be noticed that $x_1(t)$ is *directly* observable but not reachable and $x_2(t)$ is *directly* controllable. In $\mathcal{S}'$, with the introduction of the same blue block, the system become fully observable and controllable; in particular, $x_2(t)$ become also *indirectly* observable and $x_1(t)$ become also *indirectly* controllable.

Remark — 4 sub-systems.
Any State-Space system can be **internally** seen as 4 sub-systems as follows:



Which **externally** is equivalent to a systems like this:



Hence, this is graphically proves of the fact that the I/O representation of a system through the Transfer Function can only represent the *observable* and *controllable* part of the system; all the other sub-systems remains **hidden** with this representation. For this reason, the SS representation is "more complete" than the TF one when the system is not fully observable or not fully controllable.

Another final definition is needed before presenting the 4SID algorithm.

Definition 1.6 — Hankel matrix of order n. Starting from $IR = \{\omega(1), \omega(2), \dots, \omega(N)\}$ we can build the Hankel matrix of order n.

$$H_n = \begin{bmatrix} \omega(1) & \omega(2) & \omega(3) & \cdots & \omega(n) \\ \omega(2) & \omega(3) & \omega(4) & \cdots & \omega(n+1) \\ \omega(3) & \omega(4) & \omega(5) & \cdots & \omega(n+2) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \omega(n) & \omega(n+1) & \omega(n+2) & \dots & \omega(2n-1) \end{bmatrix}$$

Note: it is a square matrix of size $n \times n$.

Note: we need IR up to time $2n-1$.

Note: it starts from $\omega(1)$ and not from $\omega(0)$ (since, for strictly-proper systems, $\omega(0) = 0$).

Note: the anti-diagonals all have the same element repeated.

We know that $\omega(t) = \begin{cases} 0 & \text{if } t = 0 \\ HF^{t-1}G & \text{if } t > 0 \end{cases}$, therefore H_n can be rewritten as

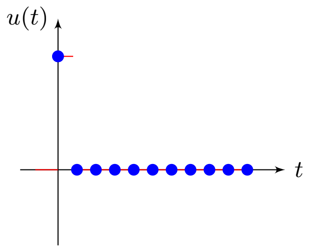
$$H_n = \begin{bmatrix} HG & HFG & HF^2G & \cdots & HF^{n-1}G \\ \vdots & \ddots & & & \vdots \\ \vdots & & \ddots & & \vdots \\ \vdots & & & \ddots & \vdots \\ HF^{n-1}G & \cdots & \cdots & \cdots & HF^{2n-2}G \end{bmatrix} = \begin{bmatrix} H \\ HF \\ \vdots \\ HF^{n-1} \end{bmatrix} \cdot \begin{bmatrix} G & FG & \cdots & F^{n-1}G \end{bmatrix} = O \cdot R$$

$$\Rightarrow H_n = O \cdot R$$

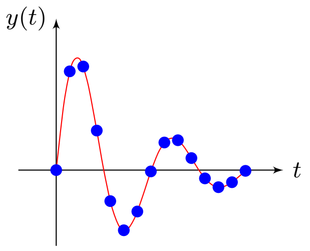
where O is the observability matrix and R is the reachability matrix.

1.4 Subspace-based State-Space System Identification (4SID)

The original 4SID method starts from the measurement of the system output in a very simple experiment, that is the *impulse experiment*.



Impulse in input



IR in output

The fundamental idea of this experiment is that it is very simple to measure the output of system given an impulse signal as the input. Since the goal of the method is to identify an SS model $\left\{\widehat{F}, \widehat{G}, \widehat{H}\right\}$ starting from the $IR = \{\omega(0), \omega(1), \omega(2), \cdots\}$, it can be said that it is a Black-Box system identification method.

We will see the solution of the problem with two different assumptions:

Basic problem IR measurement is assumed to be **noise-free**. Easier and not realistic problem. Described in 1.5.

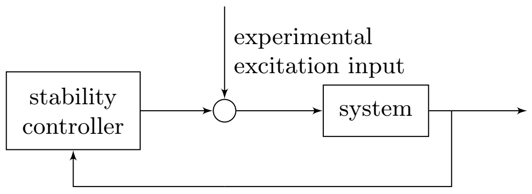
Real problem IR is measured with noise

$$\widetilde{\omega}(t) = \omega(t) + \eta(t) \quad t = 0, 1, \ldots, N$$

where $\widetilde{\omega}(t)$ is the measured noisy IR, $\omega(t)$ is the "true" noise-free IR and $\eta(t)$ is the measurement noise (e.g. White Noise (WN)). Described in 1.6.

Remark. We will see in detail only the original version of 4SID, that is that the experiment is an impulse-experiment. However 4SID can be extended to any generic input signal $\{u(1), u(2), \cdots, u(N)\}$ that is sufficiently exciting.

Remark — Unstable System. In case of an unstable system the measurements must be collected in a closed-loop experiment. Indeed, if the experiment was open-loop, the experiment would be unfeasible.



1.5 4SID procedure (noise-free)

Step 1 Build the Hankel matrix in increasing order and each time compute the rank of the matrix.

$$H_1 = \begin{bmatrix}\omega(1)\end{bmatrix} \qquad H_2 = \begin{bmatrix}\omega(1) & \omega(2)\\ \omega(2) & \omega(3)\end{bmatrix} \qquad H_3 = \dots \qquad \cdots \qquad H_n = \dots$$

Suppose that $\text{rank}(H_i) = i \quad \forall i \in \{1, \dots, n\}$ and $\text{rank}(H_{n+1}) = n$.
If this happens, it means that we found the first Hankel matrix which in not full rank and it also means that we have estimated (found) the order of the system.

Step 2 Take H_{n+1} (**recall**: $\text{rank}(H_{n+1}) = n$) and factorize it in two rectangular matrices of size $(n+1) \times n$ and $n \times (n+1)$.

$$H_{n+1} = \begin{bmatrix}\text{extended}\\ \text{observability}\\ \text{matrix:}\\ O_{n+1}\end{bmatrix} \cdot \begin{bmatrix}\text{extended}\\ \text{controllability}\\ \text{matrix:}\\ R_{n+1}\end{bmatrix}$$

where $O_{n+1} = \begin{bmatrix} H \\ HF \\ \vdots \\ HF^n \end{bmatrix}$ of size $(n+1)\times n$ and $R_{n+1} = \begin{bmatrix} G & FG & \cdots & F^nG\end{bmatrix}$ of size $n\times(n+1)$.

Step 3 H, F, G estimation.

Using O_{n+1} and R_{n+1} we can easily find:

$$\widehat{G} = R_{n+1}(:;1) \quad \text{(first column of } R_{n+1})$$
$$\widehat{H} = O_{n+1}(1;:) \quad \text{(first row of } O_{n+1})$$

To estimate $\widehat{F}$ consider O_{n+1} (or, similarly, R_{n+1}):
define O_1 as O_{n+1} without the last row, and O_2 as O_{n+1} without the first row, that is

$$O_1 = O_{n+1}\texttt{(1:n;:)} \qquad O_2 = O_{n+1}\texttt{(2:n+1;:)}$$

Note: O_1 and O_2 are $n \times n$ matrices. This is called *shift-invariance* property.

Note: O_1 is invertible since it's an observability matrix and the subsystem is fully observable.

Moreover $O_2 = O_1 F$, so $\widehat{F} = O_1^{-1} O_2$.

In conclusion in a simple and *constructive* (i.e NON parametric) way we have estimated a State-Space model of the system starting from measured *noise-free* IR, using only $2n+1$ samples of IR.
In practice $n \ll N$, where, again, n is the order of the system and N the number of sample in the dataset.

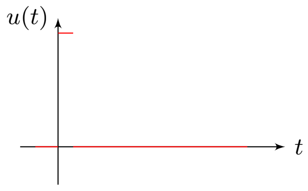
Conclusion

$$\left\{\widehat{H}, \widehat{G}, \widehat{F}\right\} = \left\{O_{n+1}\texttt{(1;:)}, R_{n+1}\texttt{(:;1)}, O_1^{-1}O_2\right\}$$

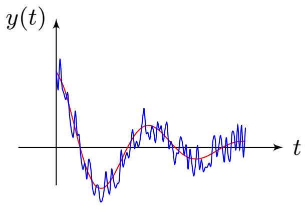
Remark. If the measurement is noisy all this process is **useless**. Also, if n is unknown, Step 1 could never stop.

1.6 4SID procedure (with Noise)

In reality, the measurements of IR are noisy: $\widetilde{\omega}(t) = \omega(t) + \eta(t)$. Therefore, the **real problem** is the estimation of the system taking into account the noise; indeed, the available dataset will be $IR = \widetilde{\omega}(t) = \{\widetilde{\omega}(0), \widetilde{\omega}(1), \cdots, \widetilde{\omega}(N)\}$.



Impulse in input



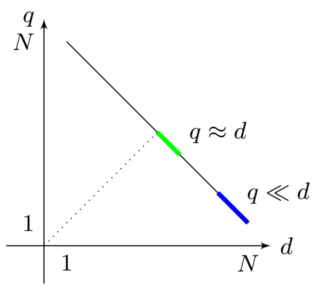
IR

Step 1 Build the Hankel matrix from data using "one-shot" all the available N data points.

$$\widetilde{H}_{qd} = \begin{bmatrix} \widetilde{\omega}(1) & \widetilde{\omega}(2) & \cdots & \widetilde{\omega}(d) \\ \widetilde{\omega}(2) & \widetilde{\omega}(3) & \cdots & \widetilde{\omega}(d+1) \\ \vdots & \vdots & \ddots & \vdots \\ \widetilde{\omega}(q) & \widetilde{\omega}(q+1) & \cdots & \widetilde{\omega}(q+d-1) \end{bmatrix}$$

$\widetilde{H}_{qd}$ is a $q \times d$ matrix.
Note that, as before, we start from $\widetilde{\omega}(1)$ since we are assuming $\widetilde{\omega}(0) = 0$.
Note that $q + d - 1$ must equal to N so that we use all the dataset.

Remark — Choice of q and d. Hypothesis: $q < d$ so that $q+d-1 = N \Rightarrow q = N+1-d$.



If $q \approx d$ the method has better accuracy but high computational effort.

If $q \ll d$ it's computationally less intensive but has the worst accuracy.

Rule of thumb: If $q > \frac{d}{2}$ we get to a good enough result.

Step 2 *Singular Value Decomposition (SVD)* of $\widetilde{H}_{qd}$

$$\underbrace{\widetilde{H}_{qd}}_{q\times d} = \underbrace{\widetilde{U}}_{q\times q}\underbrace{\widetilde{S}}_{q\times d}\underbrace{\widetilde{V}^{\mathrm{T}}}_{d\times d}$$

where $\widetilde{U}$ and $\widetilde{V}$ are *square* and *unitary* matrices and $\widetilde{S}$ is a *rectangular diagonal* matrix such that:

$$\widetilde{S} = \begin{bmatrix} \sigma_1 & & & & \\ & \sigma_2 & & & \\ & & \ddots & & \\ & & & \sigma_d & \end{bmatrix}$$

where $\sigma_1, \sigma_2, \ldots, \sigma_q$ are the *singular values* of $\widetilde{H}_{qd}$. Those are real, positive numbers, sorted in decreasing order (i.e. $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_q$).

Definition 1.7 — Unitary matrix. A matrix M is *unitary* if:

- $\det M = 1$ (so it's invertible)
- $M^{-1} = M^{\mathrm{T}}$

Observation. The singular values of a rectangular matrix are a *sort of eigenvalues* of a square matrix.
SVD is a *sort of diagonalization* of a rectangular matrix.

Recall. For a square matrix, $\mathrm{eig}(A) = \mathrm{roots}(\det(A - \lambda I))$. If M is rectangular, $SV(M) = \sqrt{\mathrm{eig}(MM^{\mathrm{T}})}$ (for non zero eigenvalues).

Remark — How to compute SVD. The optimal numerical computation is not trivial.

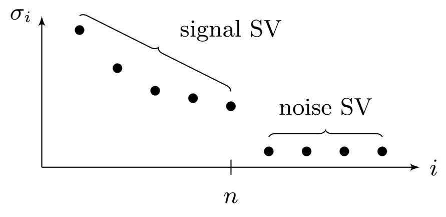
Theoretical method for SVD computation is to make 2 diagonalization steps:

$$\underbrace{\widetilde{H}_{qd}\widetilde{H}_{qd}^{\mathrm{T}}}_{q\times q} = \widetilde{U}\widetilde{S}\widetilde{S}^{\mathrm{T}}\widetilde{U}^{\mathrm{T}}$$

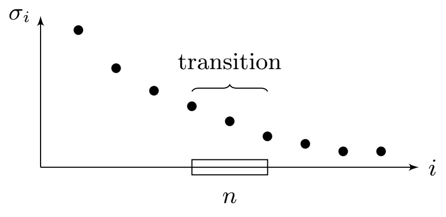
$$\underbrace{\widetilde{H}_{qd}^{\mathrm{T}}\widetilde{H}_{qd}}_{d\times d} = \widetilde{V}\widetilde{S}^{\mathrm{T}}\widetilde{S}\widetilde{V}^{\mathrm{T}}$$

Instead, use `svd(M)` in Matlab.

Step 3 Plot the singular values and separate (cut-off) the system from noise.



Ideal case

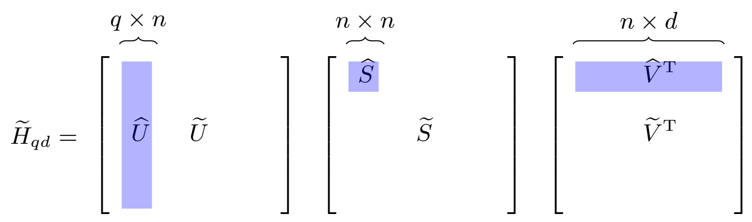


Real case

In the ideal case there is a perfect/clear separation between the signal and the noise S.V. (a jump). The index of the jump is n, that is the order of the system.

In the real case there is no clear distinction between signal and noise singular values, since the order of the system can assume values in an interval. With some empirical test we can select a good compromise between complexity, precision and overfitting (see *cross-validation*).

After the decision on the value of n we split $\widetilde{H}_{qd}$ in $\widetilde{U}$, $\widetilde{S}$ and $\widetilde{V}^{\mathrm{T}}$:



$$\widetilde{H}_{qd} = \underbrace{\widehat{U}\widehat{S}\widehat{V}^{\mathrm{T}}}_{\widehat{H}_{qd}} + H_{res,qd} \qquad \mathrm{rank}(\widetilde{H}_{qd}) = q \quad \mathrm{rank}(\widehat{H}_{qd}) = n \quad \mathrm{rank}(H_{res,qd}) = q$$

where $\widehat{H}_{qd}$ is the *"signal part"* and $H_{res,qd}$ is the *"noise (or residual) part"* of $\widetilde{H}_{qd}$.

Note: $\widehat{S}$ is now a square diagonal matrix of $\sigma_1, \dots, \sigma_q$.
Note: from $\widetilde{H}_{qd}$ to $\widehat{H}_{qd}$ the rank is hugely reduced.
Note: $\widehat{H}_{qd}$ is the "cleaned" Hankel matrix.

Step 4 Estimation of $\widehat{F}$, $\widehat{G}$ and $\widehat{H}$ using the cleaned matrix $\widehat{H}_{qd}$

$$\widehat{H}_{qd} = \widehat{U}\widehat{S}\widehat{V}^{\mathrm{T}} = \widehat{U}\widehat{S}^{\frac{1}{2}}\widehat{S}^{\frac{1}{2}}\widehat{V}^{\mathrm{T}}$$

where $\widehat{S}^{\frac{1}{2}}$ is the square diagonal matrix with elements the square roots of the elements of $\widehat{S}$, that is $\widehat{S}^{\frac{1}{2}} = diag(\sqrt{\sigma_1}, \dots, \sqrt{\sigma_q})$.

Defining $\widehat{O} = \widehat{U}\widehat{S}^{\frac{1}{2}}$ and $\widehat{R} = \widehat{S}^{\frac{1}{2}}\widehat{V}^{\mathrm{T}}$, $\widehat{H}_{qd}$ become

$$\widehat{H}_{qd} = \underbrace{\widehat{O}}_{q\times n}\underbrace{\widehat{R}}_{n\times d}$$

We can view $\widehat{O}$ as the *extended observability matrix* and the $\widehat{R}$ the *extended reachability matrix* of the system.

Now it is possible to estimate $\widehat{H}$ with the first row of $\widehat{O}$ and $\widehat{G}$ with the first column of $\widehat{R}$, similarly to the Step 3 of the noise-free case.

$$\begin{aligned} \widehat{G} &= \widehat{R}(\texttt{:;1}) \quad \text{(first column of } \widehat{R}) \\ \widehat{H} &= \widehat{O}(\texttt{1;:}) \quad \text{(first row of } \widehat{O}) \end{aligned}$$

What about the estimation of $\widehat{F}$? Let's try to proceed again similarly to what we have done at the Step 3 of the noise-free case. Consider for example $\widehat{O}$ and define $\widehat{O}_1$ as $\widehat{O}$ without the last row, and $\widehat{O}_2$ as $\widehat{O}$ without the first row.

Using the *shift-invariance* property we have that $\widehat{O}_1\widehat{F} = \widehat{O}_2$, but $\widehat{O}_1$ is not a square matrix so it's NOT invertible. To avoid this problem we can use the approximate *least-squares* solution.

Recall — **Solution of Linear Systems.** Consider a generic system $Ax = B$ with dimension $(h \times n) \cdot (n \times 1) = (h \times 1)$. We have 3 different cases:

1. $h < n$. We have less equations than variables: the system is *under determined* and we have infinite solutions.
2. $h = n$. We have one and only one solution if A is invertible.
3. $h > n$. We have more equations than variables: the system is *over determined* and it's impossible (no solutions).

Remark — **Least-Squares Method.** In case we have more equations than variable (i.e. $h > n$) we can use an approximate solution using the least-squares method, which for a generic system is as follows:

$$Ax = B$$
$$\Downarrow$$

$$A^{\mathrm{T}}Ax = A^{\mathrm{T}}B \implies \widehat{X} = \underbrace{(A^{\mathrm{T}}A)^{-1}A^{\mathrm{T}}}_{A^+} B$$

A^+ is called *pseudo-inverse*, which is “surrogate inverse” when A is rectangular.

Applying the least-square method to $\widehat{O}_1\widehat{F} = \widehat{O}_2$ we have

$$\widehat{O}_1\widehat{F} = \widehat{O}_2 \quad \Rightarrow \quad \widehat{O}_1^{\mathrm{T}}\widehat{O}_1\widehat{F} = \widehat{O}_1^{\mathrm{T}}\widehat{O}_2 \quad \Rightarrow \quad \widehat{F} = \left(\widehat{O}_1^{\mathrm{T}}\widehat{O}_1\right)^{-1}\widehat{O}_1^{\mathrm{T}}\widehat{O}_2$$

Conclusion Starting from a noisy IR $\{\widetilde{\omega}(1), \widetilde{\omega}(2), \dots, \widetilde{\omega}(N)\}$ we have estimated a model $\{\widehat{F}, \widehat{G}, \widehat{H}\}$ in a non-parametric and constructive way.

$$\left\{\widehat{H}, \widehat{G}, \widehat{F}\right\} = \left\{\widehat{O}(\mathtt{1;:}), \widehat{R}(\mathtt{:;1}), \left(\widehat{O}_1^{\mathrm{T}}\widehat{O}_1\right)^{-1}\widehat{O}_1^{\mathrm{T}}\widehat{O}_2\right\}$$

Observation. This method can be extended also to the case where the input signal is generic (i.e. not an impulse).

Remark — Optimality of 4SID. The method is *optimal* in the sense that it makes the best possible rank reduction of $\widetilde{H}_{qd}$, that is from q to n.

However, notice that in general there are infinite ways to make a rank reduction.

Remark — Rank Reduction. Our goal is to obtain the desired rank reduction by discarding the minimum amount of information contained in the original matrix. SVD makes exactly this: $\widetilde{H}_{res,qd}$ is the minimum possible (in the sense of the *Frobenius norm*).

$$\left|\widetilde{H}_{res,qd}\right|_F = \sqrt{\sum_{ij}\left(\widetilde{H}_{res,qd}^{(ij)}\right)^2}$$

Example — Rank Reduction.

$$\underbrace{\begin{bmatrix} 2 & 5 & 3 & 6 & 5 \\ 5 & 3 & 6 & 5 & 7 \\ 3 & 6 & 5 & 7 & 1 \end{bmatrix}}_{\text{rank=3}} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}}_{\text{rank=2}} + \underbrace{\begin{bmatrix} 1 & 5 & 3 & 6 & 5 \\ 5 & 2 & 6 & 5 & 7 \\ 3 & 6 & 5 & 7 & 1 \end{bmatrix}}_{\text{rank=3}}$$

It’s not the optimal rank reduction matrix, because it factors out a matrix with lower rank but a lot of information of the original matrix is lost.

Remark. 4SID is a constructive method that can be implemented in a fully-automatic way, except for these steps:

- q and d selection (not critical)

- choice of n (typically supervised by the designer). It can be made automatic using a cross-validation method.

Remark. SVD was an historical turning point in machine learning algorithms because it allows:

- very efficient compression of information.
- very efficient separation of *important* information from noise.
- order reduction of a model.

Notice that those are 3 different prospective of the same general problem.

Example — similar to an exam exercise. Consider the following S.S. model

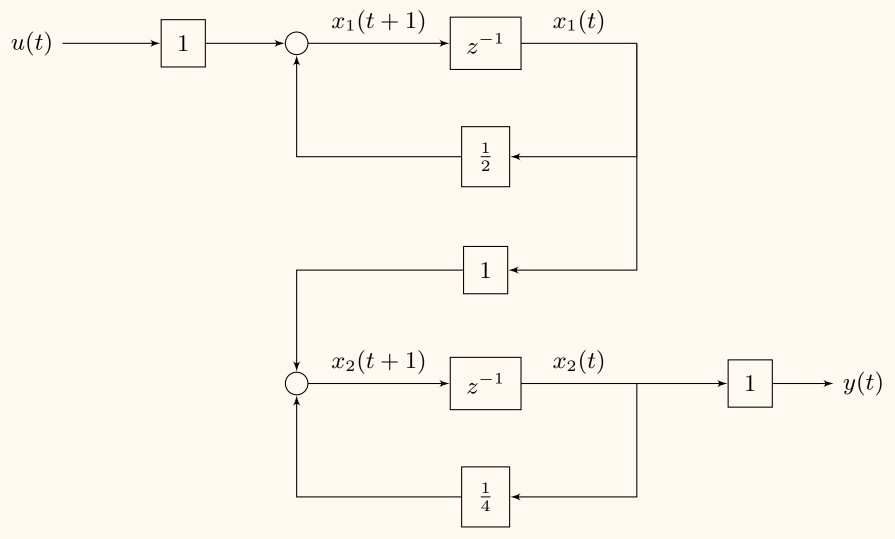
$$F = \begin{bmatrix} \frac{1}{2} & 0 \\ 1 & \frac{1}{4} \end{bmatrix} \qquad G = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \qquad H = \begin{bmatrix} 0 & 1 \end{bmatrix} \qquad D = 0$$

Note Given the size of F the system has grade $n = 2$ and it is strictly-proper single-input single-output system. **Note** Since F is triangular, the eigenvalues are on the diagonal and $\text{eig}(F) = \{\frac{1}{2}, \frac{1}{4}\}$ and they have both absolute value less than one, thus the system is *asymptotically stable.*

Question 1 Write the time domain equations of the system in the state space representation.

$$\mathcal{S} : \begin{cases} x_1(t+1) & = \frac{1}{2}x_1(t) + u(t) \\ x_2(t+1) & = x_1(t) + \frac{1}{4}x_2(t) \\ y(t) & = x_2(t) \end{cases}$$

Question 2 Write the block scheme of the SS representation of the system.



By visual inspection *seems* that the system is fully observable and fully controllable.

Question 3 Make a formal verification that the system is fully observable and controllable.

$$O = \begin{bmatrix} H \\ HF \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & \frac{1}{4} \end{bmatrix} \qquad \text{rank}(O) = 2 = n$$

$$R = \begin{bmatrix} G & FG \end{bmatrix} = \begin{bmatrix} 1 & \frac{1}{2} \\ 0 & 1 \end{bmatrix} \qquad \text{rank}(R) = 2 = n$$

This confirms our visual inspection.

Let's compute the extended $(n+1 = 2+1)$ O_3 and R_3:

$$F^2 = FF = \begin{bmatrix} \frac{1}{2} & 0 \\ 1 & \frac{1}{4} \end{bmatrix} \begin{bmatrix} \frac{1}{2} & 0 \\ 1 & \frac{1}{4} \end{bmatrix} = \begin{bmatrix} \frac{1}{4} & 0 \\ \frac{3}{4} & \frac{1}{16} \end{bmatrix}$$

$$O_3 = \begin{bmatrix} H \\ HF \\ HF^2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & \frac{1}{4} \\ \frac{3}{4} & \frac{1}{16} \end{bmatrix}$$

$$R_3 = \begin{bmatrix} G & FG & F^2G \end{bmatrix} = \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{4} \\ 0 & 1 & \frac{3}{4} \end{bmatrix}$$

Question 4 Compute the transfer function representation.

First method: direct manipulation of SS equations

$$\mathcal{S} : \begin{cases} x_1(t+1) & = \frac{1}{2}x_1(t) + u(t) \\ x_2(t+1) & = x_1(t) + \frac{1}{4}x_2(t) \\ y(t) & = x_2(t) \end{cases}$$

$$zx_1(t+1) - \frac{1}{2}x_1(t) = u(t) \qquad \implies \qquad x_1(t) = \frac{1}{z - \frac{1}{2}}u(t)$$

$$zx_2(t) - \frac{1}{4}x_2(t) = \frac{1}{z - \frac{1}{2}}u(t) \qquad \implies \qquad x_2(t) = \frac{1}{(z - \frac{1}{4})(z - \frac{1}{2})}u(t)$$

$$y(t) = \frac{1}{(z - \frac{1}{4})(z - \frac{1}{2})}u(t)$$

$$W(z) = \frac{1}{(z - \frac{1}{4})(z - \frac{1}{2})}$$

There are 2 poles: $z = \frac{1}{4}$ and $z = \frac{1}{2}$. Since the system is fully observable and controllable the poles correspond to the eigenvalues of F.

Second method: use the formula

$$W(z) = H(zI - F)^{-1}G = \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} z - \frac{1}{2} & 0 \\ -1 & z - \frac{1}{4} \end{bmatrix}^{-1} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \frac{1}{(z - \frac{1}{4})(z - \frac{1}{2})}$$

Question 5 Write I/O time-domain representation

$$y(t) = \frac{1}{z^2 - \frac{3}{4}z + \frac{1}{8}}u(t) = \frac{z^{-2}}{1 - \frac{3}{4}z^{-1} + \frac{1}{8}z^{-2}}u(t) = \frac{3}{4}y(t-1) - \frac{1}{8}y(t-2) + u(t-2)$$

Question 6 Compute the first 6 values (including $\omega(0)$) of IR.

We decide to compute it from the TF $\frac{z^{-2}}{1-\frac{3}{4}z^{-1}+\frac{1}{8}z^{-2}}$. Performing the 5-step long division of $W(z)$ the result is $z^{-2} + \frac{3}{4}z^{-3} + \frac{7}{16}z^{-4} + \frac{15}{64}z^{-5}$.

$$\omega(0) = 0 \qquad \omega(1) = 0 \qquad \omega(2) = 1$$
$$\omega(3) = \frac{3}{4} \qquad \omega(4) = \frac{7}{16} \qquad \omega(5) = \frac{15}{64}$$

Note Also $\omega(1) = \omega(0) = 0$: this means that the delay $k = 2$, not just 1.

Question 7 Build the Hankel matrix and stop when the rank is not full (noise-free case).

$$H_1 = \begin{bmatrix} 0 \end{bmatrix} \qquad \text{rank}(H_1) = 1$$
$$H_2 = \begin{bmatrix} 0 & 1 \\ 1 & \frac{3}{4} \end{bmatrix} \qquad \text{rank}(H_2) = 2$$
$$H_3 = \begin{bmatrix} 0 & 1 & \frac{3}{4} \\ 1 & \frac{3}{4} & \frac{7}{16} \\ \frac{3}{4} & \frac{7}{16} & \frac{15}{64} \end{bmatrix} \qquad \text{rank}(H_3) = 2 \neq 3$$

H_3 is not full rank so the order of the system is 2 (as we already know).

It can be proved that $O_3R_3 = H_3$.

Chapter 2

Parametric Black-Box system identification of I/O system using a frequency domain approach

So far we have seen:

- In MIDA1 parametric black-box identification of I/O systems (ARMAX) and time series (ARMA)
- In Chapter 1 non-parametric black-box identification of I/O systems (4SID)

The **frequency domain approach** is a black-box and **parametric**, and it's very used in practice since it's very robust and reliable.

Since it's parametric it uses the 4 usual steps:

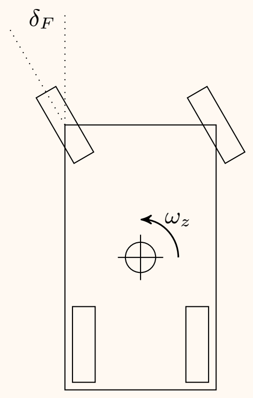
1. Experiment design and data pre-processing. A special type of experiment and data pre-processing is needed.
2. Selection of parametric model class ($\mathcal{M}(\vartheta)$)
3. Definition of a performance index ($J(\vartheta)$). A new special performance index is needed.
4. Optimization ($\widehat{\vartheta} = \arg\min_{\vartheta} J(\vartheta)$)

1. and 2. are the special steps of this chapter.

The general intuitive idea of the method is:

- Make a set of "single sinusoid" ("single-tune") excitation experiments
- From each experiment estimate a single point of the frequency response of the system
- Fit the estimated and modeled frequency response to obtain the optimal model

Example — Car steer dynamics. Let's picture a top-view of a car that is steering.



where δ_F is the *steer angle* (input) and ω_z is the *rotational speed* around the vertical axis z.

This kind of dynamics relationship is very important for stability control systems design (ESG/ESP) and for autonomous cars.

There are 3 possible situations:

$$\begin{tikzcd}\delta_F \arrow[r] & \boxed{1} \arrow[r] & \omega_z\end{tikzcd}$$

$$\begin{tikzcd} & \delta_F \arrow[d] & \\ \delta_R \arrow[r] & \boxed{2} \arrow[r] & \omega_z\end{tikzcd}$$

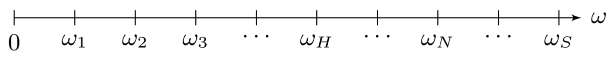
$$\begin{tikzcd}\delta_F \arrow[dr] & & \\ & \boxed{3} \arrow[r] & \omega_z \\ \delta_R \arrow[ur] & & \end{tikzcd}$$

1. The control variable is the *front steer* δ_F. This is the case of autonomous cars
2. The human driver controls δ_F which is a measurable disturbance while the system controls the *rear steer* δ_R
3. Both δ_R and δ_F are control variables: application high performance autonomous car

Imagine to make an experiment where you make a sinusoid of the steer and you get the rotational speed as output.

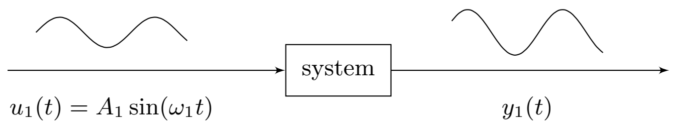
2.1 Steps of the system identification with frequency domain method

Step 1 In the experiment design step we first have to select a set of *excitation frequencies*.

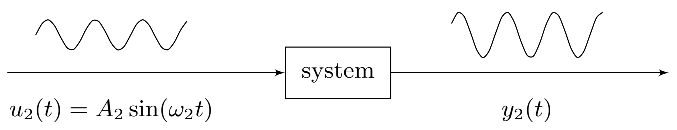


where ω_S is the *sampling frequency*, $\omega_N = \frac{1}{2}\omega_S$ is the *Nyquist frequency* (which is the maximum frequency of a digital system) and ω_H is the maximum explored frequency. We have $\{\omega_1, \omega_2, \cdots, \omega_H\}$ usually evenly spaced ($\Delta\omega$ is constant). ω_H must be selected according to the bandwidth of the control system.

We make H *independent* experiments.

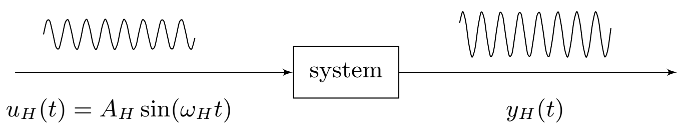


Experiment #1



Experiment #2

$\vdots$



Experiment #H

Remark. The amplitudes A_1, A_2, …, A_H can be equal (constant) or, more frequently in practice, they decrease as the frequency increases to fulfill the power constraint on the input.

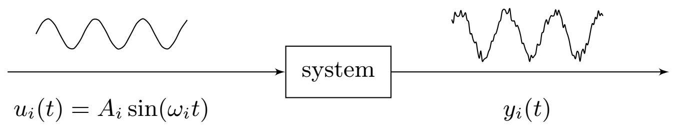
Indeed, if $\delta(t)$ is the steering angle (moved by an actuator), the requested steer torque is proportional to δ: $T(t) = K\delta(t)$. Therefore the steer power is proportional to $T(t)\dot{\delta}(t) = K\delta(t)\dot{\delta}(t)$. If $\delta(t) = A_i \sin(\omega_i t)$ the steering power is $KA_i \sin(\omega_i t)\omega_i A_i \cos(\omega_i t)$ which is proportional to $KA_i^2\omega_i$.

If we have a limit to this power, this power should be constant during the H experiments, thus

$$KA_i^2\omega_i = \text{const} \qquad \Longrightarrow \qquad A_i = \sqrt{\frac{\text{const}}{K\omega_i}}$$

and this means that the amplitude must be inversely proportional to the frequency.

Focusing on the i-th experiment

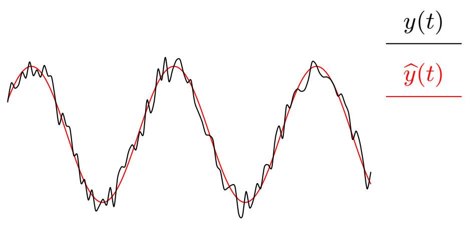


Recall — Frequency Response Theorem for LTI Systems. If the system is LTI (linear time-invariant), the frequency response theorem says that if the input is a sine input of frequency ω_i the output must be a sine with frequency ω_i but with different amplitude or phase.

However $y_i(t)$ in real applications is not a perfect sinusoid because of those non-ideal behaviours:

- Noise on output measurements
- Noise on the system not directly related to measurements (e.g. roughness of the road).
- (Small) non-linear effects (that we neglect, since we will use LTI local approximations of the system)

In pre-processing of I/O data we want to extract from $y_i(t)$ a perfect sinusoid of frequency ω_i. We force the assumption that the system is LTI, so the output must be a pure sine wave of frequency ω_i (all the remaining signal is noise).



The model of the output signal is

$$\widehat{y}_i = B_i \sin(\omega_i t + \phi_i) = a_i \sin(\omega_i t) + b_i \cos(\omega_i t)$$

There are 2 unknowns: B_i and ϕ_i (or a_i and b_i). It's better to use the model with a_i and b_i since it's *linear* in those parameters.

The unknown parameters are a_i and b_i and we can find them by parametric identification.

$$\{\widehat{a}_i, \widehat{b}_i\} = \underset{\{a_i, b_i\}}{\arg\min}\, J_N(a_i, b_i)$$

$$J_N(a_i, b_i) = \frac{1}{N}\sum_{t=1}^{N} (y_i(t) - \widehat{y}_i(t))^2 = \frac{1}{N}\sum_{t=1}^{N}\left(\underbrace{y_i(t)}_{\text{measurement}} \underbrace{-a_i \sin(\omega_i t) - b_i \cos(\omega_i t)}_{\text{model output}}\right)^2$$

Since the model is linear in a_i and b_i, J_N, which is the *sample variance of the modelling error*, is a *quadratic function* of a_i and b_i. Thus, we can solve the problem explicitly.

$$\frac{\partial J_N}{\partial a_i} = \frac{2}{N}\sum_{t=1}^{N}(-\sin(\omega_i t))(y_i(t) - a_i \sin(\omega_i t) - b_i \cos(\omega_i t)) = 0$$
$$\frac{\partial J_N}{\partial b_i} = \frac{2}{N}\sum_{t=1}^{N}(-\cos(\omega_i t))(y_i(t) - a_i \sin(\omega_i t) - b_i \cos(\omega_i t)) = 0$$

This is a 2×2 linear system which can be written in matrix form

$$\begin{bmatrix} \sum_{t=1}^{N} \sin(\omega_i t)^2 & \sum_{t=1}^{N} \sin(\omega_i t)\cos(\omega_i t) \\ \sum_{t=1}^{N} \sin(\omega_i t)\cos(\omega_i t) & \sum_{t=1}^{N} \cos(\omega_i t)^2 \end{bmatrix} \begin{bmatrix} a_i \\ b_i \end{bmatrix} = \begin{bmatrix} \sum_{t=1}^{N} y_i(t)\sin(\omega_i t) \\ \sum_{t=1}^{N} y_i(t)\cos(\omega_i t) \end{bmatrix}$$

At this point we prefer to go back to a *sin-only* form (B_i, ϕ_i) of the estimated sinusoid $\widehat{y}_i(t)$:

$$\widehat{B}_i \sin(\omega_i t + \phi_i) = \widehat{B}_i \sin(\omega_i t)\cos(\widehat{\phi}_i) + \widehat{B}_i \cos(\omega_i t)\sin(\widehat{\phi}_i) \overset{!}{=} \widehat{a}_i \sin(\omega_i t) + \widehat{b}_i \cos(\omega_i t)$$

$$\Downarrow$$

$$\begin{cases} \widehat{B}_i \cos(\widehat{\phi}_i) = \widehat{a}_i \\ \widehat{B}_i \sin(\widehat{\phi}_i) = \widehat{b}_i \end{cases} \tag{$\bigstar$}$$

$$\Downarrow$$

$$\frac{\widehat{b}_i}{\widehat{a}_i} = \frac{\sin \widehat{\phi}_i}{\cos \widehat{\phi}_i} = \tan(\widehat{\phi}_i) \qquad \widehat{\phi}_i = \arctan\left(\frac{\widehat{b}_i}{\widehat{a}_i}\right)$$

$$\widehat{B}_i = \frac{\frac{\widehat{a}_i}{\cos \widehat{\phi}_i} + \frac{\widehat{b}_i}{\sin \widehat{\phi}_i}}{2} \quad \text{average of the the 2 equation in } (\bigstar)$$

Therefore, we have now found a bidirectional mapping $\{\widehat{a}_i, \widehat{b}_i\} \iff \{\widehat{B}_i, \widehat{\phi}_i\}$.
Repeating the experiment and pre-processing for the H experiments we can compute

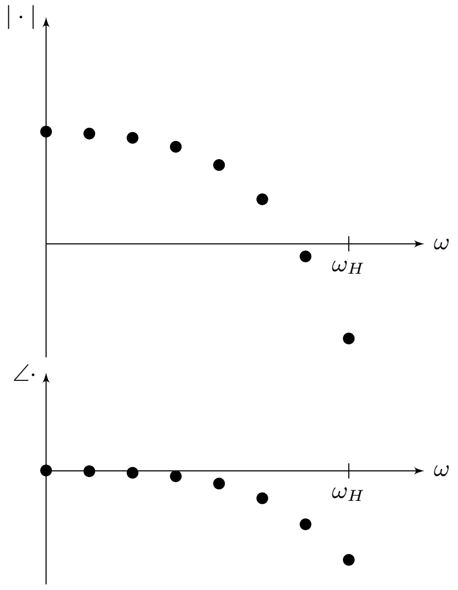
$$\begin{aligned} \{\widehat{B}_1, \widehat{\phi}_1\} &\mapsto \frac{\widehat{B}_1}{A_1} e^{j\widehat{\phi}_1} \\ \{\widehat{B}_2, \widehat{\phi}_2\} &\mapsto \frac{\widehat{B}_2}{A_2} e^{j\widehat{\phi}_2} \\ &\vdots \\ \{\widehat{B}_H, \widehat{\phi}_H\} &\mapsto \frac{\widehat{B}_H}{A_H} e^{j\widehat{\phi}_H} \end{aligned}$$

where each $\frac{\widehat{B}_i}{A_i} e^{j\widehat{\phi}_i}$ is a complex number that is the estimated point at frequency ω_i of the frequency response of the transfer function $W(z)$ from the input $u(t)$ to the output $y(t)$ of the system.

> **Recall — Frequency Response Theorem for LTI systems.** If for a LTI system characterized by TF $W(z)$ the input is $A_i \sin(\omega_i t)$ and the output is $\widehat{B}_i \sin(\omega_i t + \widehat{\phi}_i)$, then $W(z = e^{j\omega_i}) = \frac{\widehat{B}_i}{A_i} e^{j\widehat{\phi}_i}$ is the corresponding frequency response of the system at that frequency (ω_i).

In particular, $\frac{\widehat{B}_i}{A_i}$ is the ratio between the output and input amplitude and $\widehat{\phi}_i$ is the phase shift.

We can now plot these H point in a classical *Bode plot* exploiting that $|W(e^{j\omega_i})| = \frac{\widehat{B}_i}{A_i}$ and that $\angle W(e^{j\omega_i}) = \widehat{\phi}_i$.



At the end of *step 1* we have a frequency-domain dataset (H values) representing H estimated points of the frequency response of the system.

Step 2 Parametric model class (TF) selection

$$\mathcal{M}(\vartheta) : W(z;\vartheta) = \frac{b_0 + b_1 z^{-1} + \cdots + b_p z^{-p}}{1 + a_1 z^{-1} + \cdots + a_n z^{-n}} z^{-1} \qquad \vartheta = \begin{bmatrix} a_1 \\ \vdots \\ a_n \\ b_0 \\ \vdots \\ b_p \end{bmatrix}$$

Observation — **Model order selection.** In this case the order is composed by 2 parameters n and p: use cross-validation approach (or visual fitting in the Bode diagram) for finding the best choice of these parameters.

Step 3 New performance index: variance of the error in frequency domain.

$$J_H(\vartheta) = \frac{1}{H} \sum_{i=1}^{H} \left| W(e^{j\omega_i};\vartheta) - \frac{\widehat{B}_i}{A_i} e^{j\widehat{\phi}_i} \right|^2$$

where $W(e^{j\omega_i};\vartheta)$ is the modeled frequency response and $\frac{\widehat{B}_i}{A_i}e^{j\widehat{\phi}_i}$ is the measured frequency response.

Step 4 Optimization

$$\widehat{\vartheta}_H = \arg\min_{\vartheta} J_H(\vartheta)$$

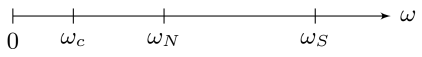
Usually $J_H(\vartheta)$ is a *non-quadratic* and *non-convex function*; Thus, iterative optimization methods are needed.

Conclusion We have obtained the estimated model which is a TF

$$\mathcal{M}(\widehat{\vartheta}_H) : W(z;\widehat{\vartheta}_H)$$

Remark — Frequency bandwidth selection $\omega_H = ?$. Theoretically the standard best solution should be H points distributed uniformly from 0 to ω_N (Nyquist freq.).

In practice it's better to concentrate the experimental effort in a smaller and more focused bandwidth.

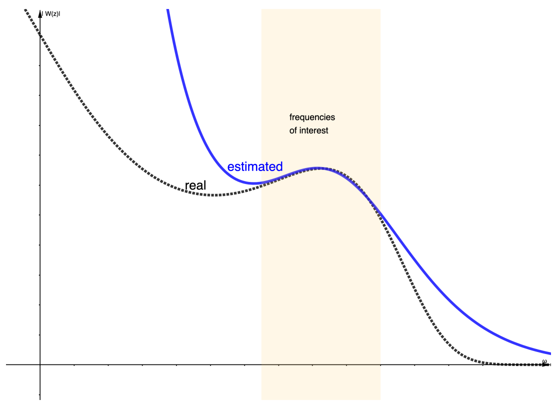


where ω_c is the *expected cut-off frequency* of the closed system.

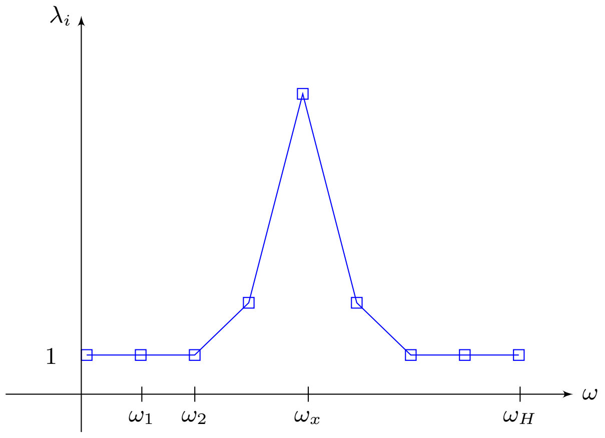
Rule of thumb: $\omega_H \approx 3\omega_c$

Example The Electronic Stability Control (ESC) system has an expected bandwidth of $\omega_c \approx 4$Hz, so $\omega_H \approx 12$Hz.

Remark — Emphasis on special frequency range. In some cases, between ω_1 and ω_H, we want to be more accurate in system identification in some frequency regions (typically around cut-off-frequency or around resonances).



We can manage this selected focus on estimation precision using non-uniform weights λ_i (different weights for different frequencies).



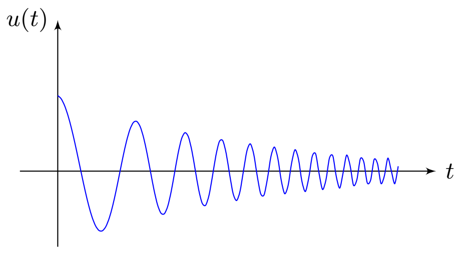
where ω_x is the frequency of interest.

The performance index can be redefined:

$$\widetilde{J}_H(\vartheta) = \frac{1}{H} \sum_{i=1}^{H} \lambda_i \left(W(e^{j\omega_i};\vartheta) - \frac{\widehat{B}_i}{A_i} e^{j\widehat{\phi}_i} \right)^2$$

Another *trick*: more dense ω_i spacing in the frequency region of special interest (not really used).

Remark — **Single experiment.** Sometimes the set of H independent single-sinusoid experiments can be replaced by a long single "sine-sweep" experiment, that is to measure the frequency response of the system when the input is sinusoid that starts with frequency ω_1 and amplitude A_1 and ends with frequency ω_H and amplitude A_H.



Slowing-varying sinusoid with increasing frequency and decreasing amplitude.

The output will be a single long signal $y(t)$.

We can cut a-posteriori the signal into H pieces, and then back to the standard procedure or we can directly compute an estimation of $\widehat{W}(e^{j\omega})$ as a ration of the output and input spectra

$$\widehat{W}(e^{j\omega}) = \frac{\widehat{\Gamma}_y(e^{j\omega})}{\widehat{\Gamma}_u(e^{j\omega})}$$

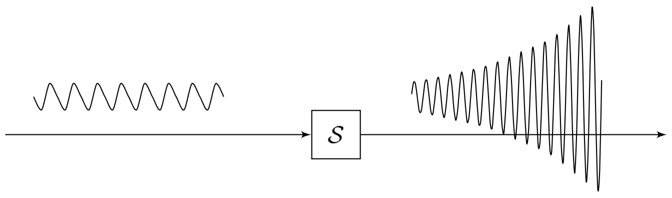
We can fit the estimated $\widehat{W}(e^{j\omega})$ with the model frequency response $W(e^{j\omega};\vartheta)$ in the performance index.

$$\frac{1}{H}\sum_{i=1}^{H}\left|W(e^{j\omega_i};\vartheta) - \widehat{W}(e^{j\omega_i})\right|^2$$

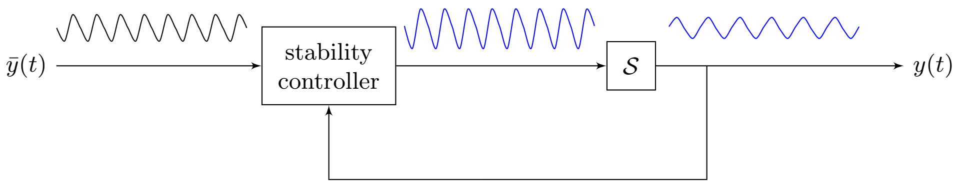
where, similarly, $W(e^{j\omega_i};\vartheta)$ is the modeled frequency response and $\widehat{W}(e^{j\omega_i})$ is the measured frequency response.

This experiment is quicker but has usually a lower signal-to-noise-ration.

Remark — Experiment on unstable system. What happen if the system is *open-loop* unstable? We have to stop the experiment because of the instability of the output (which of course diverges).



We can avoid this problem by performing the experiment in *closed-loop* and adding a *stability controller.*



where the blue sinusoids are the signal reflecting the dynamic of the system and, measuring them, they will compose the dataset I/O of the system.

Note that the model identified using that dataset will be a model of an unstable system.

This is just a trick to collect data from an unstable system.

2.2 Comparison between time domain and frequency domain parametric methods

Frequency domains

Pro Robust and very reliable because each experiment has a big signal-to-noise ratio (since we are forcing all the signal energy on a single clean sinusoid).

Pro Intuitive since it is easy to understand.

Pro Consistent with many control-design methods that work in the frequency domain.

Cons More demanding in terms of design of the experiment.

Cons Provides no noise model (unlike the PEM method of ARMAX system identification)

Note that F.D. and T.D. methods should provide approximately the same result if done correctly.

Chapter 3

Software (virtual) Sensing model-based in feedback method using Kalman Filter framework

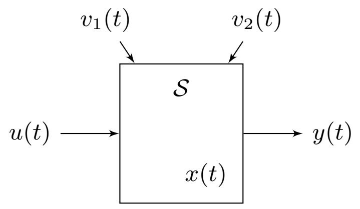
In MIDA1 we have mostly used I/O Transfer Function (TF) representations:

$$y(t) = \frac{B(z)}{A(z)}u(t-k) + \frac{C(z)}{A(z)}e(t) \qquad e(t) \sim \text{WN}$$

Kalman filter theory is fully based on State-Space (SS) representation

$$\begin{cases} x(t+1) = Fx(t) + Gu(t) + v_1(t) & \qquad v_1 \sim \text{WN} \\ y(t) = Hx(t) + \cancel{Du(t)} + v_2(t) & \qquad v_2 \sim \text{WN} \end{cases}$$

where we assume that the system model is given (typically obtained in a white-box approach): it's not a system identification problem! Indeed we are interested in the internal state of the system.



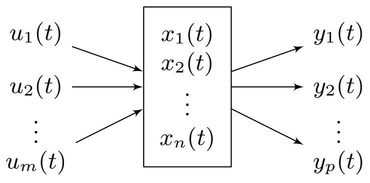
Two-noises model

3.1 Motivation and Goals of Kalman Filter

Given a model description $\{F, G, H\}$ and noises variances (not a system identification technique), with Kalman Filter (KF) theory we can address the following problems:

- Find the k-steps ahead prediction of output: $\widehat{y}(t+k|t)$ (already solved in MIDA1 with ARMAX).
- Find the k-steps ahead prediction of state: $\widehat{x}(t+k|t)$.
- Find the filter of the state at present time: $\widehat{x}(t|t)$. In practice it's a *software-sensing* problem, which is the most important problem solved by Kalman filter (reason of why it is named Kalman *filter*).
- Gray box system identification (see **??**). We have a recorded data-set and the model structure with some unknown parameters.

Dynamical systems have this layout:



MIMO system with m inputs (actuators/control variables), p outputs (sensors) and n states.

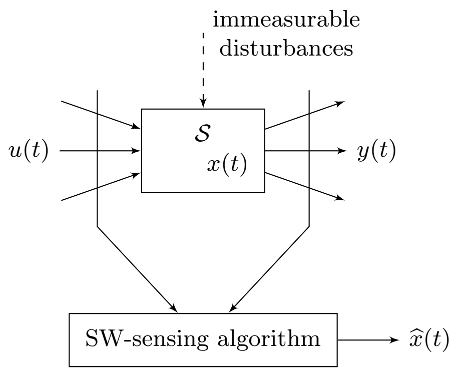
Key problem usually $p \ll n$, that is, physical sensors are much less than system states, because of:

- cost
- cables, power supply, installation
- maintenance (faults, degradation)
- existence of such sensors

That's why usually not all the states are measured (available) but it is very useful to have full "measurement" of states because:

- design of control algorithms (state feedback design)
- monitoring of the system (fault detection, predictive maintenance, ...)

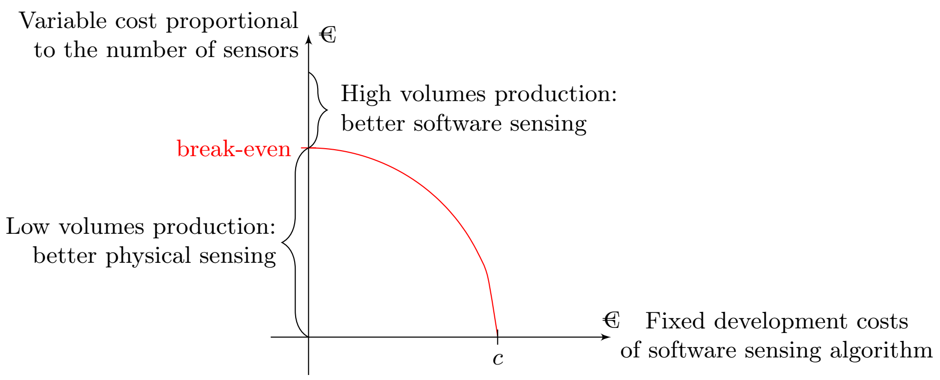
This problem can be solved with *software sensing (SW-sensing)* (also called virtual sensing) algorithms:



where $\widehat{x}(t)$ is an estimation (or SW-sensing) of the internal state $x(t)$.

Designer Dilemma When using software sensing and when physical sensing?

- In some cases there is no option (not feasible installation of a physical sensor)
- In most cases both options are viable: variable vs fixed cost



Break-even analysis

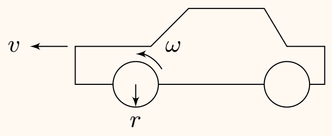
Given c, the fixed cost of development of the software sensor, it is possible to compute the break-event point. The break-even point coincides with the number of products such that above that number it is more convenient to use software sensing, while below that number it is more convenient to buy and install physical sensors per each product.

Anyway, in some cases we might use both physical and software sensing for redundancy (e.g. in vert safety-critical or mission-critical applications).

Key questions for software sensing

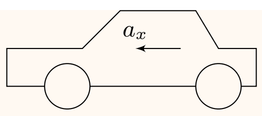
- Is software-sensing feasible? Test is the observability of the states from measured outputs (through physical sensors).
- If feasible, check if the level of noise (error) of the estimated variable is acceptable.

EXAMPLE — **Slip estimation for ABS/traction control.** Now we go back to the ABS example (described in 0.3) where we realized that SW-sensing is needed for the estimation of v, the horizontal velocity of the car, since it cannot be measured with a physical sensor.



Indeed, measure of v cannot be done with an optical sensor or a GPS: they both have a problem of availability (not guaranteed). Physical sensing is not an option for industrial production.

Intuitive solution: install a longitudinal accelerometer (a_x) and integrate.



$$\widehat{v} = \int a_x(t)dt \qquad \Longleftrightarrow \qquad a_x(t) \rightarrow 1/s \rightarrow \widehat{v}(t)$$

In discrete time domain: discretization using approximation of derivative, Eulero forward method (see A)

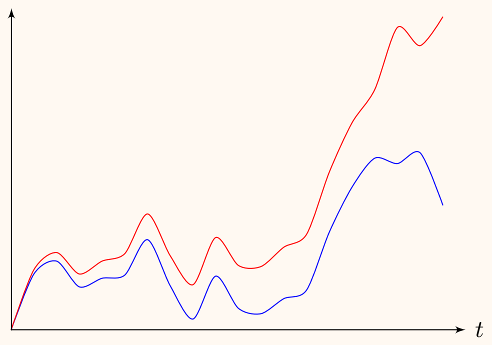
$$\frac{d}{dt}v(t) = a_x(t) \qquad \frac{dv(t)}{dt} \approx \frac{v(t+1)-v(t)}{\Delta T_s} = a_x(t)$$

Where ΔT_s is the sampling interval (e.g. 10ms).

$$\widehat{v}(t) = \widehat{v}(t-1) + \Delta T_s a_x(t-1)$$

$$\begin{tikzcd} {} \arrow[r, "a_x(t)"] & \boxed{\frac{\Delta T_s}{1-z^{-1}}} \arrow[r, "\widehat{v}(t)"] & {} \end{tikzcd}$$

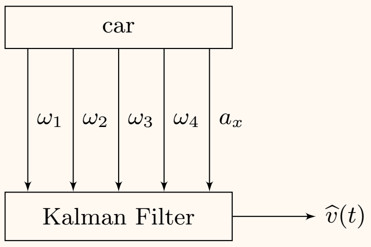
Unfortunately the measured signal is not $a_x(t)$ but $a_x(t) + d_{a_x}(t)$.

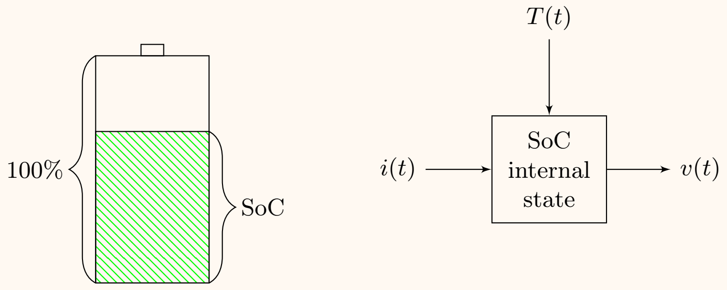


Integrating noise generates a *drift*. Integrator is not an asymptotic stable system.

$$\begin{tikzcd} d_{a_x}(t) \arrow[d, "+"'] & & \\ \bigcirc \arrow[r] & \boxed{\frac{\Delta T_s}{1-z^{-1}}} \arrow[r, "\widehat{v}(t)"] & {} \\ a_x(t) \arrow[u, "+"] & & \end{tikzcd}$$

Solution Use a Kalman Filter.





Example — State of charge estimation of a battery.

$$\text{SoC}(t) = 1 - \frac{\int i(t)dt}{I} \qquad 0 \le \text{SoC} \le 1$$

Where I is the total amount of *current* that can be extracted by the user of the battery. This solution is not feasible since it integrates the noise on $i(t)$.

3.2 Kalman Filter on Basic Systems

- No external inputs ($\cancel{Gu(t)}$): time series
- Linear systems
- Time invariant systems

The basic solution of this basic system is the 1-step prediction.
After that, we will make the extensions to more general systems:

- k-step prediction
- filter $\widehat{x}(t|t)$
- time-varying systems
- systems with exogenous inputs (presence of $Gu(t)$)
- non-linear systems (Extended Kalman Filter (EKF))

3.2.1 Detailed description of Basic System

The basic system we initially consider is a MIMO system with n states, (m inputs) and p outputs.

$$\mathcal{S}: \begin{cases} x(t+1) = Fx(t) + \cancel{Gu(t)} + v_1(t) & \text{state equation} \\ y(t) = Hx(t) + \cancel{Du(t)} + v_2(t) & \text{output equation} \end{cases}$$

$$x(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \\ \vdots \\ x_n(t) \end{bmatrix} \qquad \left(u(t) = \begin{bmatrix} u_1(t) \\ u_2(t) \\ \vdots \\ u_m(t) \end{bmatrix} \right) \qquad y(t) = \begin{bmatrix} y_1(t) \\ y_2(t) \\ \vdots \\ y_p(t) \end{bmatrix}$$

State Noise $v_1(t)$ is a vector white-noise.

$$v_1(t) \sim \mathrm{WN}(0, V_1) \qquad v_1(t) = \begin{bmatrix} v_{11}(t) \\ v_{12}(t) \\ \vdots \\ v_{1n}(t) \end{bmatrix}$$

and it is called *state noise* or *model noise*. It is used to model immeasurable noises affecting the system and small modelling errors.

Properties of $v_1(t)$:

1. $\mathbb{E}[v_1(t)] = \vec{0}$
2. $\mathbb{E}[v_1(t) \cdot v_1^{\mathrm{T}}(t)] = V_1$, where V_1 is an $n \times n$ covariance matrix (square, symmetric and semi-definite positive by definition)
3. $\mathbb{E}[v_1(t) \cdot v_1^{\mathrm{T}}(t-\tau)] = 0 \quad \forall t \quad \forall \tau \neq 0$ (*whiteness* property)

Output Noise $v_2(t)$ is a vector white-noise.

$$v_2(t) \sim \mathrm{WN}(0, V_2) \qquad v_2(t) = \begin{bmatrix} v_{21}(t) \\ v_{22}(t) \\ \vdots \\ v_{2p}(t) \end{bmatrix}$$

and it is called *output noise* or *sensor noise*. It is the noise affecting the output sensor measurements.

Properties of $v_2(t)$:

1. $\mathbb{E}[v_2(t)] = \vec{0}$
2. $\mathbb{E}[v_2(t) \cdot v_2^{\mathrm{T}}(t)] = V_2$, where V_2 is a $p \times p$ covariance matrix (square, symmetric and semi-definite positive[1] by definition)
3. $\mathbb{E}[v_2(t) \cdot v_2^{\mathrm{T}}(t-\tau)] = 0 \quad \forall t \quad \forall \tau \neq 0$ (*whiteness* property)

[1]We make the **assumption that V_2 is definite positive** (i.e. $V_2 > 0$) because we will need this property in the Riccati equation.

Assumptions about the relationships between $v_1(t)$ and $v_2(t)$:

$$\mathbb{E}[v_1(t) \cdot v_2^{\mathrm{T}}(t-\tau)] = \begin{cases} 0 & \text{if } \tau \neq 0 \\ V_{12} & \text{if } \tau = 0 \end{cases} \qquad \clubsuit$$

where V_{12} is a cross-correlation matrix of size $n \times p$.

The system ($\clubsuit$) means that v_1 and v_2 can be correlated only at the same time, but, in practice, $V_{12} = 0$ is the most common assumption. Thus, in practice

$$\mathbb{E}[v_1(t) \cdot v_2^{\mathrm{T}}(t-\tau)] = 0 \quad \forall t \forall \tau \qquad (1^{st} \text{ assumption})$$

Since the system $\mathcal{S}$ is dynamic we need to define the (in this case, probabilistic) initial conditions:

$$\mathbb{E}[x(1)] = \underbrace{X_0}_{n\times 1} \qquad \mathbb{E}[(x(1)-X_0))\,(x(1)-X_0)^{\mathrm{T}}] = \underbrace{P_0}_{n\times n} \geq 0$$

If the covariance matrix $P_0 = 0$ the initial state is perfectly known.

Finally we assume that the two noises $v_1(t)$ and $v_2(t)$ are uncorrelated with the initial state:

$$x(1) \perp v_1(t) \qquad x(1) \perp v_2(t) \qquad (2^{nd} \text{ assumption})$$

3.2.2 KF Basic Solution of the Basic System

State eq.: $$\widehat{x}(t+1|t) = F\widehat{x}(t|t-1) + K(t)e(t) \tag{3.1}$$

Output eq.: $$\widehat{y}(t|t-1) = H\widehat{x}(t|t-1) \tag{3.2}$$

Prediction output error eq.: $$e(t) = y(t) - \widehat{y}(t|t-1) \tag{3.3}$$

Gain of the filter: $$K(t) = \left(FP(t)H^{\mathrm{T}} + V_{12}\right)\left(HP(t)H^{\mathrm{T}} + V_2\right)^{-1} \tag{3.4}$$

DRE: $$P(t+1) = \left(FP(t)F^{\mathrm{T}} + V_1\right) - \left(FP(t)H^{\mathrm{T}} + V_{12}\right)\left(HP(t)H^{\mathrm{T}} + V_2\right)^{-1}\left(FP(t)H^{\mathrm{T}} + V_{12}\right)^{\mathrm{T}} \tag{3.5}$$

Since 3.1 and 3.5 are dynamical equations, two initial conditions are needed:

Init. condition for 3.1: $$\widehat{x}(1|0) = \mathbb{E}[x(1)] = X_0 \tag{3.6}$$

Init. condition for 3.5: $$P(1) = \mathrm{var}[x(1)] = P_0 \tag{3.7}$$

Definition 3.1 — Difference Riccati Equation (DRE). The equation 3.5 is called *Difference Riccati Equation (DRE)* and it is a special type of non-linear matrix difference equation.

Note The Difference Riccati Equation (DRE) is an autonomous (i.e. there are no inputs), non-linear, discrete time, multi-variable system, described by a non-linear difference matrix equation

$$P(t+1) = f_{NL}(P(t)) \qquad P(1) = P_0$$

Remark — Structure or $K(t)$ and DRE. Notice that $K(t)$ and the DRE have a *block-structure* having this form:

$$AP(t)B^{\mathrm{T}} + N \qquad \text{where } N \text{ is a noise matrix}$$

There are 3 different types of blocks:

$$\begin{aligned} \texttt{state:} \quad & FP(t)F^{\mathrm{T}} + V_1 \qquad \text{(since } F \text{ refers to 3.1)} \\ \texttt{output:} \quad & HP(t)H^{\mathrm{T}} + V_2 \qquad \text{(since } H \text{ refers to 3.2)} \\ \texttt{mix:} \quad & FP(t)H^{\mathrm{T}} + V_{12} \end{aligned}$$

Therefore

$$\begin{aligned} \text{3.4 become:} \quad & K(t) \equiv (\texttt{mix})(\texttt{output})^{-1} \\ \text{3.5 become:} \quad & P(t+1) \equiv (\texttt{state}) - (\texttt{mix})(\texttt{output})^{-1}(\texttt{mix})^{\mathrm{T}} \end{aligned}$$

Remark — Existance of the DRE. In order to guarantee the existance of the DRE for all time instant t, the only critical part is the inversion of the `output` block:

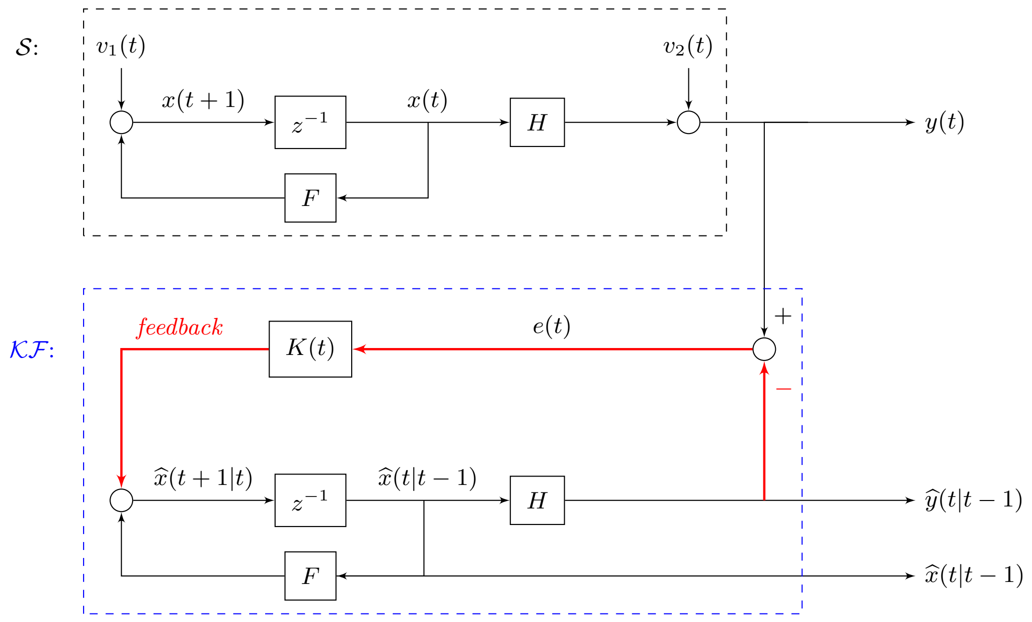
$$\underbrace{(\underbrace{HP(t)H^{\mathrm{T}}}_{\geq 0} + \underbrace{V_2}_{>0})^{-1}}_{>0} \qquad \text{thanks to } V_2 > 0, \texttt{ output} \text{ is always invertible}$$

Remark — Meaning of $P(t)$. The symmetric $n \times n$ matrix $P(t)$ has a very important meaning, indeed

$$\begin{aligned} P(t) = \mathbb{E}[(x(t) - \widehat{x}(t|t-1))(x(t) - \widehat{x}(t|t-1))^{\mathrm{T}}] &= \mathrm{Var}[x(t) - \widehat{x}(t|t-1)] \\ &= \mathrm{Var}[e_x(t)] \end{aligned}$$

Therefore, $P(t)$ is the covariance matrix of the 1-step prediction error of the state $x(t)$.

3.2.3 Block-scheme representation of the Kalman Filter



The idea behind Kalman Filter is simple and intuitive:

- we make a simulated replica (*digital twin*) of the system (without noises v_1 and v_2 which are not measurable)
- we compare the true measured output with the estimated/predicted output $\widehat{y}(t|t-1)$
- we make corrections on the KF main equation, proportional (with gain $K(t)$) to the output error $e(t)$ in order to keep KF as close as possible to the system
- we extract the state estimation $\widehat{x}(t|t-1)$ from the digital twin

Observation. Kalman Filter is a feedback system. Feedback here is not used for control, but for estimation.

This general structure was known since ′30 (before Kalman Filter development) and was called *state observer*. Fundamental contribution of Kalman was to find the **optimal gain** $K(t)$. $K(t)$ is not a simple scalar gain but is a (maybe very large) $n \times p$ matrix.

The selection of gain matrix $K(t)$ is very critical:

- if $K(t)$ is *too small*: the estimation is not optimal because we are *under exploiting* the information in $y(t)$
- if $K(t)$ is *too big*: risk of over-exploiting $y(t)$ and we can get noise amplification, even risk of instability

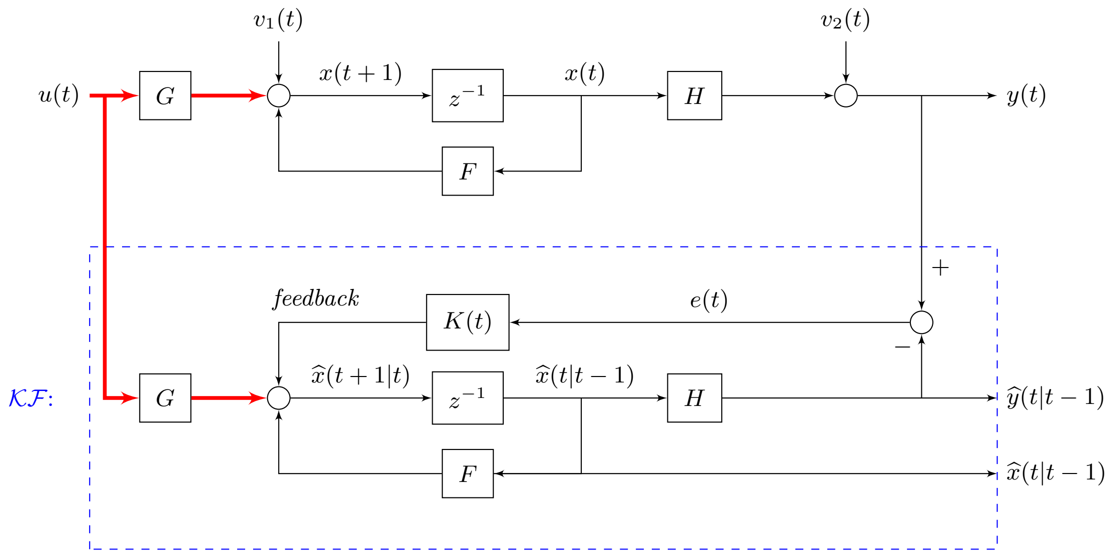
Design of a Kalman Filter does not require a *training dataset*, but a complete model of the system:

- F, G, H matrixes: usually obtained with a white-box physical modelling of the system
- V_1, V_2 and V_{12}: V_2 is easily built from sensor specifications, while V_1 is much more

difficult to be designed (it's the most critical design parameter of KF). For what concerns V_{12}, we recall that in practice $V_{12} = 0$.

3.3 Extensions of the KF for General System

3.3.1 Exogenous input



Notice that $K(t)$ remains the same because $P(t)$ is the covariance of the prediction error on $x(t)$ which remains the same because $Gu(t)$ doesn't introduce any additional noise or uncertainties to the system. That's because $Gu(t)$ is a totally known (deterministic) signal.

3.3.2 Multi-step Prediction

Assuming that $\widehat{x}(t+1|t)$ is known from the basic solution, we can simply obtain a multi-step prediction as:

$$\begin{aligned}\widehat{x}(t+2|t) &= F\widehat{x}(t+1|t)\\ \widehat{x}(t+3|t) &= F\widehat{x}(t+2|t) = F^2\widehat{x}(t+1|t)\\ &\vdots\end{aligned}$$

$$\begin{cases}\widehat{x}(t+k|t) = F^{k-1}\widehat{x}(t+1|t)\\ \widehat{y}(t+k|t) = H\widehat{x}(t+k|t)\end{cases}$$

3.3.3 Filter ($\widehat{x}(t|t)$)

$$\widehat{x}(t+1|t) = F\widehat{x}(t|t) \quad \Longrightarrow \quad \widehat{x}(t|t) = F^{-1}\widehat{x}(t+1|t)$$

This formula can be used only if F is invertible. If F is not invertible, the filter can be obtained with a specific *filter* formulation of KF

Definition 3.2 — Kalman Filter in Filter Form. Reformulation of the equations described in 3.2.2.
State equation (3.5) and Gain equation (3.4) become

$$\begin{aligned}\widehat{x}(t|t) &= F\widehat{x}(t-1|t-1) + Gu(t-1) + K_0(t)e(t)\\ K_0(t) &= \left(P(t)H^{\mathrm{T}}\right)\left(HP(t)H^{\mathrm{T}} + V_2\right)^{-1}\end{aligned}$$

while the DRE equation (3.5) remains unchanged.

The initial condition for the new State equation is

$$\widehat{x}(1|1) = X_0$$

Remark. These equations are valid under the (legit) assumption $V_{12} = 0$.

Observation. Gain of KF in prediction form (eq. 3.4 assuming $V_{12} = 0$):

$$K(t) = \left(FP(t)H^{\mathrm{T}}\right)\left(HP(t)H^{\mathrm{T}} + V_2\right)^{-1}$$

Gain of KF in filter form:

$$K_0(t) = \left(\cancel{F}P(t)H^{\mathrm{T}}\right)\left(HP(t)H^{\mathrm{T}} + V_2\right)^{-1}$$

Therefore, the only difference is the elimination of F.

3.3.4 Time-varying systems

$$\begin{aligned}F &\mapsto F(t)\\ G &\mapsto G(t)\\ H &\mapsto H(t)\end{aligned}$$

Therefore

$$\mathcal{S}: \begin{cases} x(t+1) = F(t)x(t) + G(t)u(t) + v_1(t)\\ y(t) = H(t)x(t) + v_2(t)\end{cases}$$

Definition 3.3 — Linear Time Varying (LTV) system. The system $\mathcal{S}$ is a *Linear Time Varying (LTV)* system, since its parameters (F, G, H) depends on the time instant t.

Kalman Filter equations remain of the same form as the ones described in 3.2.2: we just need to replace the parameter matrices (F, G, H) with the time-varying ones $(F(t), G(t), H(t))$.

3.3.5 Non-Linear Systems

This extensions is much more complicated. We will see Extended Kalman Filter (EKF) in section 3.4.2.

3.4 Asymptotic Solution of Kalman Filter

Observation. Even when the system $\mathcal{S}$ is an LTI, the Kalman Filter is not itself an LTI system: it is an LTV system, since it depends on the gain $K(t)$ which is time-varying.

The fact that KF is an LTV system is the source of 2 problems:

- Checking the asymptotic stability of KF algorithm is very difficult, since the stability check of an LTV system is not simple as the stability check for LTI.
- Computational problem: $K(t)$ must be computed at each sampling time (e.g. every 5ms), including the inversion of $HP(t)H^{\mathrm{T}} + V_2$ ($p \times p$ matrix) and the computation of $P(t)$ using the DRE.

Recall — Asymptotic Stability of a system. If the system is:

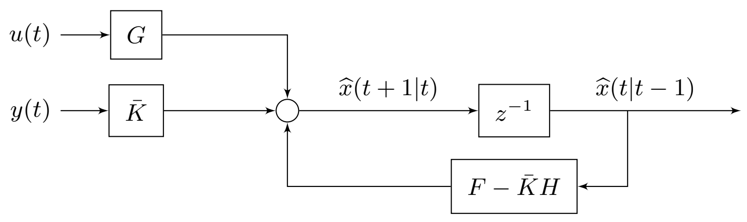
- LTI: $x(t+1) = Fx(t) + Gu(t)$
  the stability check considers only the sign of the eigenvalues of F.
- LTV: $x(t+1) = F(t)x(t) + G(t)u(t)$
  even if all the eigenvalues of $F(t)$ are strictly inside the unit circle at any time, the system is not guaranteed asymptotically stable. In practice it is, if the time-variations are *slow* (e.g. aging).

Because of those problems in real/practical applications the asymptotic version of KF is preferred.

Basic idea Since the dependency on t of $K(t)$ derives from the presence of $P(t)$, if $P(t)$ converges to a constant value $\bar{P}$ (steady-state value of $P(t)$), then also $K(t)$ will converge to $\bar{K}$ (steady-state value of $K(t)$).
Using $\bar{K}$ the KF becomes an LTI system.

Let's analyze the asymptotic stability of the State equation (eq. 3.1) of the asymptotic KF when $\bar{K}$ is used (assuming it exists).

$$\begin{aligned}
\widehat{x}(t+1|t) &= F\widehat{x}(t|t-1) + Gu(t) + \bar{K}e(t) \\
&= F\widehat{x}(t|t-1) + Gu(t) + \bar{K}(y(t) - \widehat{y}(t|t-1)) \\
&= F\widehat{x}(t|t-1) + Gu(t) + \bar{K}(y(t) - H\widehat{x}(t|t-1)) \\
&= \underbrace{(F - \bar{K}H)}_{\text{new state matrix}} \widehat{x}(t|t-1) + Gu(t) + \bar{K}y(t)
\end{aligned}$$



Asymptotic Kalman Filter with Exogenous input

Condition for Asymptotic Stability of KF If $\bar{K}$ exists, the KF is asymptotically stable if and only if all the eigenvalues of $F-\bar{K}H$ are strictly inside the unit circle.

Observation. The stability of the system $\mathcal{S}$ is related to matrix F, whereas the stability of KF is related to matrix $F-\bar{K}H$.

Therefore, KF can be asymptotically stable even if the system is unstable.

Existance of $\bar{K}$ Starting from the equation 3.4, if exists $P(t)$ such that $P(t)=\bar{P}$ then $K(t)$ becomes

$$\bar{K}=\left(F\bar{P}H^{\mathrm{T}}+V_{12}\right)\left(H\bar{P}H^{\mathrm{T}}+V_2\right)^{-1}$$

Thus, $\bar{K}$ exists if $\bar{P}$ exists.

In order to check that, we need to check the converge properties of DRE

Recall — Stability of a dynamical autonomous system. How to find the equilibrium points of a dynamical autonomous system?

Continuous time	**Discrete time**
$\dot{x}=f(x(t))$	$x(t+1)=f(x(t))$
equilibrium when $\dot{x}=0$ $\downarrow$ $f(\bar{x})=0$	equilibrium when $x(t+1)=x(t)$ $\downarrow$ $f(\bar{x})=\bar{x}$

DRE is an autonomous discrete time system, thus we impose $\bar{P}=f(\bar{P})$, where $f(\bar{P})$ is the DRE (eq. 3.5) evaluated in $\bar{P}$:

$$\text{ARE:}\quad \bar{P}=\left(F\bar{P}F^{\mathrm{T}}+V_1\right)-\left(F\bar{P}H^{\mathrm{T}}+V_{12}\right)\left(H\bar{P}H^{\mathrm{T}}+V_2\right)^{-1}\left(F\bar{P}H^{\mathrm{T}}+V_{12}\right)^{\mathrm{T}} \tag{3.8}$$

Definition 3.4 — Algebraic Riccati Equation (ARE). The equation 3.8 is a non-linear, matrix, static algebraic equation, known as *Algebraic Riccati Equation (ARE).*

If a steady state $\bar{P}$ solution of DRE (eq. 3.5) does exists, it must be a solution of A.R.E (eq. 3.8). There remains 3 questions:

1. **Existence**: does Algebraic Riccati Equation (ARE) have a semi-definite positive solution?
2. **Convergence**: if exists, does the DRE converges to $\bar{P}$?
3. **Stability**: is the corresponding $\bar{K}$ such that the KF is asymptotically stable?

To answer those questions we need two fundamental theorems (KF asymptotic theorems).

3.4.1 Asymptotic Kalman Filter Theorems

The two Asymptotic Kalman Filter Theorems that we are going to introduce provide *sufficient* conditions only.

Theorem 3.5 — First Asymptotic KF Theorem. Assumptions: $V_{12} = 0$ and the system is asymptotically stable (i.e. all eigenvalues of F are strictly inside the unit circle). Then:

1. ARE has one and only one semi-definite positive solution: $\bar{P} \geq 0$.
2. DRE converges to $\bar{P}$, $\forall P_0 \geq 0$ (P_0: initial semi-definite positive condition).
3. The corresponding $\bar{K}$ is s.t. the KF is asymptotically stable (i.e. all the eigenvalues of $F - \bar{K}H$ have absolute value less than 1).

Recall — Observability and Controllability. Recall on Observability and Controllability needed for the introduction of the Second Asymptotic KF Theorem (3.6).

Observability of the state through the output The pair (F, H) is observable if and only if

$$O = \begin{bmatrix} H \\ HF \\ \vdots \\ HF^{n-1} \end{bmatrix} \quad \text{is full rank}$$

Controllability from the noise We are interested in controllability from $v_1(t)$ (and not from $u(t)$).

$$x(t+1) = Fx(t) + \cancel{Gu(t)} + v_1(t) \qquad v_1(t) \sim WN(0, V_1)$$

It's always possible to factorize $V_1 = \Gamma \cdot \Gamma^T$ rewriting

$$x(t+1) = Fx(t) + \Gamma\omega(t) \qquad \omega(t) \sim WN(0, I)$$

We can say that the state x is controllable/reachable from the input noise $v_1(t)$ if and only if:

$$R = \begin{bmatrix} \Gamma & F\Gamma & \cdots & F^{n-1}\Gamma \end{bmatrix} \qquad \text{is full rank}$$

Theorem 3.6 — Second Asymptotic KF Theorem. Assumptions: $V_{12} = 0$, (F, H) is observable and (F, Γ) is controllable. Then:

1. ARE has one and only one definite positive solution $\bar{P} > 0$.
2. DRE converges to $\bar{P}$, $\forall P_0 \geq 0$ (P_0: initial semi-definite positive condition).
3. The corresponding $\bar{K}$ is such that the KF is asymptotically stable (i.e. all the eigenvalues of $F - \bar{K}H$ have absolute value less than 1).

Observation. The difference between theorems 3.5 and 3.6 is that the first one ensures that $\bar{P} \geq 0$ while the second one ensures that $\bar{P} > 0$.

These two theorems are very useful in practice because we can fully avoid the (very difficult) direct convergence analysis of DRE.

Example.

$$S : \begin{cases} x(t+1) = \frac{1}{2}x(t) + v_1(t) & v_1 \sim WN(0, \frac{19}{20}) \\ y(t) = 2x(t) + v_2(t) & v_2 \sim WN(0, 1) \end{cases} \qquad v_1 \perp v_2$$

Question Find (if exists) the steady state (asymp.) K.F. $\widehat{x}(t+1|t)$ and $\widehat{x}(t|t)$.

$$n = 1 \qquad F = \frac{1}{2} \qquad G = 0 \qquad H = 2 \qquad V_1 = \frac{19}{20} \qquad V_2 = 1 \qquad V_{12} = 0$$

Since $V_{12} = 0$ we can try to use the asymptotic theorems.

First step Compute the DRE

$$\begin{aligned} P(t+1) &= \left(FP(t)F^T + V_1\right) - \left(FP(t)H^T + V_{12}\right)\left(HP(t)H^T + V_2\right)^{-1}\left(FP(t)H^T + V_{12}\right)^T \\ &= \frac{1}{4}P(t) + \frac{19}{20} - \frac{\left(\frac{1}{\cancel{2}}P(t)\cancel{2}\right)^2}{4P(t)+1} \\ &= \frac{\cancel{P(t)^2} + \frac{1}{4}P(t) + \frac{19}{5}P(t) + \frac{19}{20} - \cancel{P(t)^2}}{4P(t)+1} \end{aligned}$$

Note The second order terms must cancel out.

$$P(t+1) = \frac{81P(t) + 19}{80P(t) + 20}$$

Second step Compute and solve the ARE

$$\overline{P} = \frac{81\overline{P}+19}{80\overline{P}+20} \quad \Longrightarrow \quad 80\overline{P}^2 + 20\overline{P} - 81\overline{P} - 19 = 0$$

$$\overline{P}_1 = 1 \qquad \cancel{\overline{P}_2 = -\frac{19}{80}} < 0$$

$\overline{P} = 1$ is the only definite positive solution of ARE

Question Does DRE converges to $\overline{P} = 1$, $\forall P_0 \geq 0$?

There are 2 methods for addressing this question:

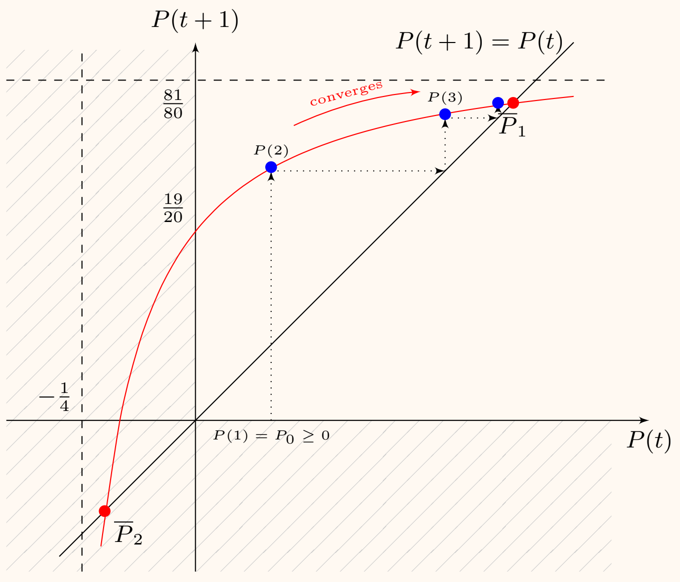
Direct analysis of DRE

- Using asymptotic theorems

First methods Direct analysis of DRE

$$P(t+1) = f(P(t)) \qquad \text{we need to plot } f(\cdot) \text{ in the } P(t) - P(t+1) \text{ plane}$$

$$\overline{P} = \frac{81\overline{P}+19}{80\overline{P}+20} \qquad \begin{cases} \text{vertical asy. value} & P(t) = -\frac{20}{80} = -\frac{1}{4} \\ \text{horizontal asy. value} & P(t) = \frac{81}{80} \end{cases}$$



By direct analysis/inspection of DRE dynamics we can conclude that $\forall P_0 \geq 0$, DRE always converges to $\overline{P}_1 = 1$.

If $n = 1$ the direct inspection is feasible, but it's very difficult for $n \geq 2$.

Second method Use theorems

$$V_{12} = 0 \qquad F = \frac{1}{2} \ (S \text{ is stable}) \qquad \Longrightarrow \qquad \text{First theorem is fulfilled}$$

The observability matrix of $\{F, H\}$ is $O = \begin{bmatrix} 2 \end{bmatrix}$ with full rank $O = 1$, the system is fully observable.

Controllability from noise $v_1(t)$

$$V_1 = \frac{19}{20} \qquad \Gamma = \sqrt{\frac{19}{20}}$$

The controllability matrix of $\{F, \Gamma\}$ is $R = \begin{bmatrix} \sqrt{\frac{19}{20}} \end{bmatrix}$ with full rank $R = 1$, the system is fully controllable from noise.

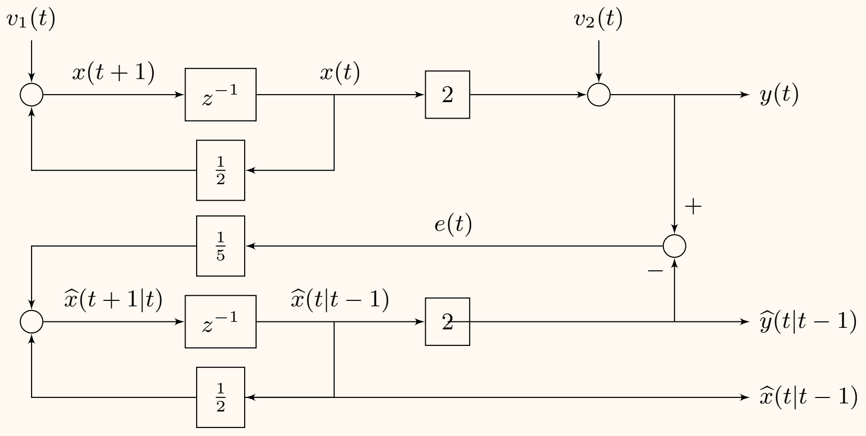
Both theorems are fulfilled, so ARE has one and only one solution $\overline{P} > 0$, DRE converges to $\overline{P}$, $\forall P_0 \geq 0$ and $\overline{K}$ makes the K.F. asymptotically stable.

Third step Compute $\overline{K}$

$$\overline{K} = \left(F\overline{P}H^T + V_{12}\right)\left(H\overline{P}H^T + V_2\right) = \left(\frac{1}{2}\cdot 1 \cdot 2 + 0\right)(2 \cdot 1 \cdot 2 + 1)^{-1} = \frac{1}{5}$$

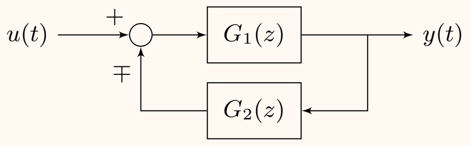
Double-check the asymptotical stability of K.F.

$$F - \overline{K}H = \frac{1}{2} - \frac{1}{5}\cdot 2 = \frac{5-4}{10} = \frac{1}{10}$$



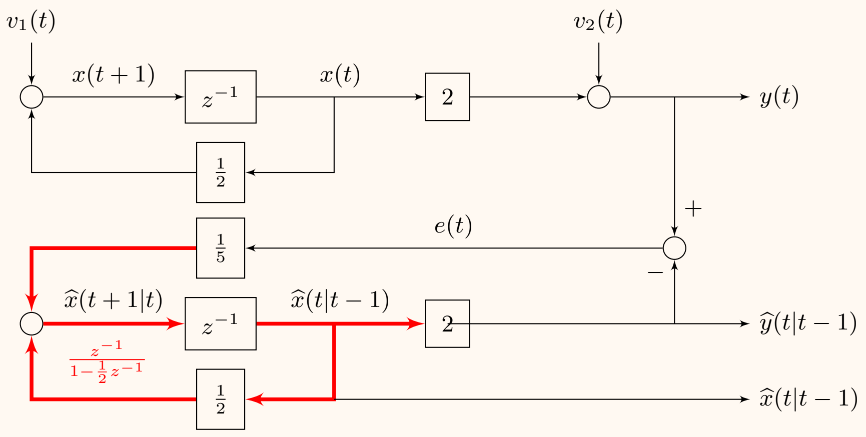
Question Find T.F. from $y(t)$ to $\widehat{x}(t|t-1)$

Recall T.F. from block schemes of feedback systems.



$$y(t) = G_1(z)\left(u(t) \mp G_2(z)y(t)\right) \implies y(t) = \frac{G_1(z)}{1 \pm G_1(z)G_2(z)}u(t)$$

The K.F. is composed of 2 nested loops.



Predictor of state:

$$\widehat{x}(t|t-1) = \frac{\frac{1}{5}\frac{z^{-1}}{1-\frac{1}{2}z^{-1}}}{1+\frac{1}{5}\frac{z^{-1}}{1-\frac{1}{2}z^{-1}}2}y(t) = \frac{\frac{1}{5}z^{-1}}{1-\frac{1}{10}z^{-1}}y(t)$$

Predictor of output:

$$\widehat{y}(t|t-1) = H\widehat{x}(t|t-1) = \frac{2}{5}\frac{1}{1-\frac{1}{10}z^{-1}}y(t-1)$$

Filter of state:

$$\widehat{x}(t|t) = F^{-1}\widehat{x}(t+1|t) = \frac{2}{5}\frac{1}{1-\frac{1}{10}z^{-1}}y(t)$$

Remark — **White noise.** In the formulas of Kalman Filter there is a requirement that $v_1(t)$ and $v_2(t)$ must be white noises. In many practical applications this assumption can be too demanding.

We need a workaround to deal with practical applications where this assumption is not valid. The workaround is a simple trick called *state extension* and it consists on extending the model of $\mathcal{S}$ to incorporate the *noise dynamics*.

Let's see how it works with a numerical example.

Example Given a system $\mathcal{S}$ of order $n = 1$

$$\mathcal{S} : \begin{cases} x(t+1) = ax(t) + \eta(t) & \eta(t) \text{ is not a white noise} \\ y(t) = bx(t) + v_2(t) & v_2(t) \sim \text{WN}(0,1) \end{cases}$$

and a model of $\eta(t)$ (AR(1) stochastic model)

$$\eta(t) = \frac{1}{1 - cz^{-1}} e(t) \qquad e(t) \sim \text{WN}(0,1) \qquad e \perp v_2$$

We cannot apply KF formula to this system, since $\eta(t)$ is not a WN, but we can proceed as follows

$$\begin{aligned} \eta(t) = c\eta(t-1) + e(t) \overset{z}{\Rightarrow} \eta(t+1) &= c\eta(t) + e(t+1) \\ \eta(t+1) &= c\eta(t) + v(t) \end{aligned}$$

where $v(t) = e(t+1)$, $\quad v \sim \text{WN}(0,1)$and $\quad v \perp v_2$.

Trick Extension of the state vector.

$$\begin{aligned} x(t) &\mapsto x_1(t) \\ \eta(t) &\mapsto x_2(t) \end{aligned}$$

Now $n = 2$ and $\mathcal{S}$ becomes $\mathcal{S}'$

$$\mathcal{S}' : \begin{cases} x_1(t+1) = ax_1(t) + x_2(t) \\ x_2(t+1) = cx_2(t) + v(t) \\ y(t) = bx_1(t) + v_2(t) \end{cases}$$

$$F \begin{bmatrix} a & 1 \\ 0 & c \end{bmatrix} \quad H = \begin{bmatrix} b & 0 \end{bmatrix} \quad v_1 = \begin{bmatrix} 0 \\ v(t) \end{bmatrix} \quad V_1 = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \quad v_2 \sim WN(0,1) \quad V_{12} = 0$$

Now we can apply KF solution equations (3.2.2) to this system.

3.4.2 Extension to Non-Linear systems

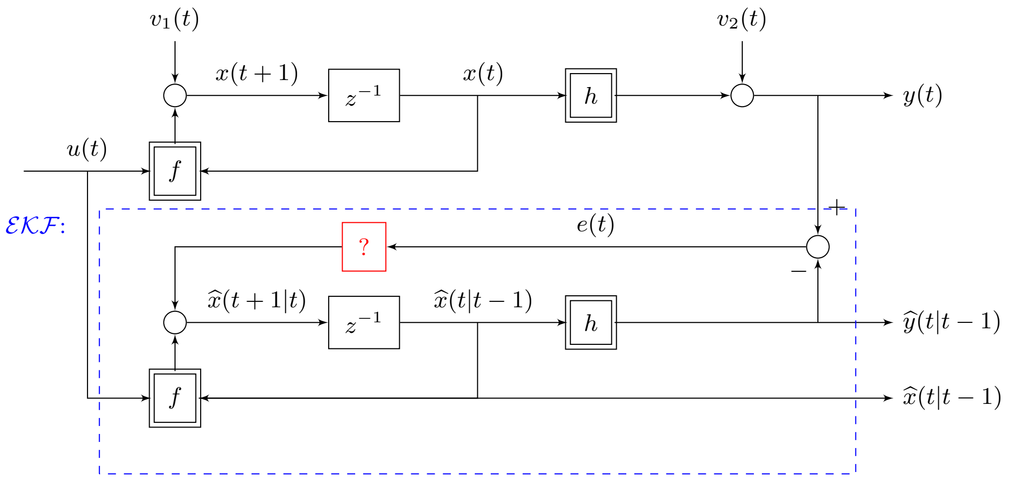
Coming back to extensions of the KF for the basic system (section 3.3), we consider a system with non-linear dynamics:

$$S : \begin{cases} x(t+1) = f(x(t), u(t)) + v_1(t) \\ y(t) = h(x(t)) + v_2(t) \end{cases}$$

where f and h are non-linear functions of $x(t)$ and $u(t)$ (smoothness class C^1 or higher), for example

$$f(x(t), u(t)) = \frac{1}{2}x^5(t) + u^3(t), \qquad h(x(t)) = e^{x(t)} + v_2(t)$$

How can we design a Kalman Filter in this case? We can follow the general idea of the *state observer*.



For the gain block of KF we have 2 different types of solutions:

1. The gain is a non-linear function of $e(t)$ (most natural and intuitive solution)
2. The gain is a linear time-varying function

The second solution is less intuitive but is the most effective: we can reuse most of the KF theory's formulas for the LTV domain.

In practice, Extended Kalman Filter (EKF) idea is to make a time-varying local linearization (approximation) of the non-linear time-invariant system at each sampling time.

The gain equation $K(t)$ in Extended Kalman Filter can be computed as:

$$K(t) = \left(F(t)P(t)H(t)^T + V_{12}\right)\left(H(t)P(t)H(t)^T + V_2\right)^{-1}$$

and $P(t)$ can be computed from the DRE:

$$P(t+1) = \left(F(t)P(t)F(t)^T + V_1\right) - \left(F(t)P(t)H(t)^T + V_{12}\right)\left(H(t)P(t)H(t)^T + V_2\right)^{-1}\left(F(t)P(t)H(t)^T + V_{12}\right)^T$$

Observation. Equations of K and DRE are the usual formulas of KF (3.2.2) with the introduction of time-varying parameters (as said in 3.3.4).

Linearization of the system We linearize f and h by computing at each sampling time, respectively, $F(t)$ and $H(t)$ as:

$$F(t) = \left.\frac{\partial f(x(t), u(t))}{\partial x(t)}\right|_{x(t)=\widehat{x}(t|t-1)}$$

$$H(t) = \left. \frac{\partial h(x(t))}{\partial x(t)} \right|_{x(t)=\widehat{x}(t|t-1)}$$

Therefore, EKF is the time-varying solution of KF where $F(t)$ and $H(t)$ are local linearized matrices computed around the last available state prediction $\widehat{x}(t|t-1)$.

Summary Procedure to implement EKF at time t:

1. take last available state prediction $\widehat{x}(t|t-1)$
2. using $\widehat{x}(t|t-1)$, linearize the system by computing $F(t)$ and $H(t)$
3. compute $K(t)$ and update the DRE
4. compute $\widehat{x}(t+1|t)$

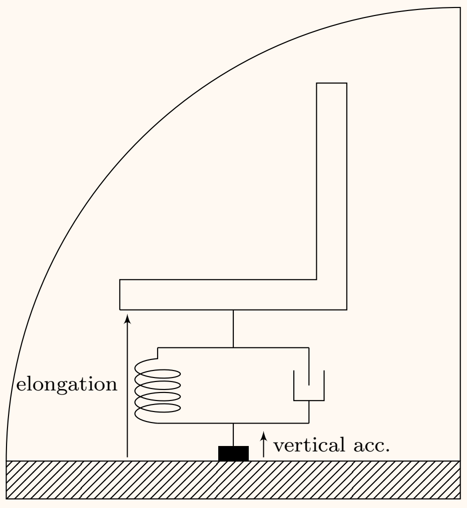
Remark. Main issues of EKF (same of LTV KF):

- Very difficult (almost impossible) to have a theoretical guarantee of EKF stability (in practice extensive empirical testing is used).
- Computational load (at each time $F(t)$, $H(t)$, $K(t)$ and $P(t)$ must be computed at run-time).

However, EKF is largely used today with some limitations in:

- safety-critical applications
- mission-critical applications

Example — KF full procedure. Suspended seat in the cabin of an off-highway vehicle (agriculture tractor, earth-moving machine, etc...).

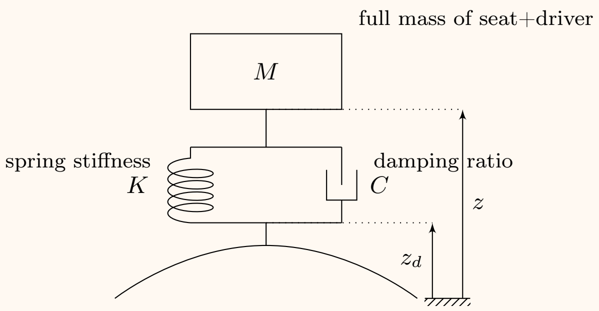


The physical sensors are:

- Vertical accelerometer placed at the basis of the cabin
- Elongation sensor between the basis of the cabin and the seat

Problem Estimation of the seat vertical speed.

Step 1: System Modeling Move from a pictorial representation to a schematic representation of it:



The sensor are:

- Acceleration $\ddot{z}_d$ (+noise)
- Elongation $z - z_d$ (+noise)

Model of the system dynamics (physical white-box model in continuous time domain).

Core model equation (force balance in vertical direction):

$$\begin{aligned} M\ddot{z} &= \underbrace{-C\frac{d}{dt}(z - z_d)}_{\text{damping}} \underbrace{-K(z - z_d)}_{spring} \cancel{+\text{gravity} - Mg} \\ &= -C(\dot{z} - \dot{z}_d) - K(z - z_d) \end{aligned}$$

Since we measure $\ddot{z}_d$ the overall dimension of the system is 4. The vector of state variables is:

$$x(t) = \begin{bmatrix} z \\ \dot{z} \\ z_d \\ \dot{z}_d \end{bmatrix} = \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \\ x_4(t) \end{bmatrix} \qquad u(t) = \ddot{z}_d \qquad y(t) = z - z_d$$

We can write the full model in state space form

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = -\frac{C}{M}(x_2 - x_4) - \frac{K}{M}(x_1 - x_3) \\ \dot{x}_3 = x_4 \\ \dot{x}_4 = u \\ y = x_1 - x_3 \end{cases} \Rightarrow \begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = -\frac{K}{M}x_1 - \frac{C}{M}x_2 + \frac{K}{M}x_3 + \frac{C}{M}x_4 \\ \dot{x}_3 = x_4 \\ \dot{x}_4 = u \\ y = x_1 - x_3 \end{cases}$$

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\frac{K}{M} & -\frac{C}{M} & \frac{K}{M} & \frac{C}{M} \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad B = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \quad C = \begin{bmatrix} 1 & 0 & -1 & 0 \end{bmatrix}$$

Step 2: Discretization Next is discretization (we use digital systems), choice of sampling time Δ (for this application can be 5ms).

Using *Eulero-forward approximation* (see A) of time derivative $\dot{x}(t) \approx \frac{x(t+1)-x(t)}{\Delta}$.

Example for the first equation:

$$\frac{x_1(t+1)-x_1(t)}{\Delta} = x_2(t) \quad \Rightarrow \quad x_1(t+1) = x_1(t) + \Delta x_2(t)$$

Step 3: Add Noises In order to apply KF formulas we need to add noises v_1, v_2 to the model. The discretized system with noises is:

$$\mathcal{S}: \begin{cases} x_1(t+1) = x_1(t) + \Delta x_2(t) + v_{11}(t) \\ x_2(t+1) = -\frac{\Delta K}{M} x_1(t) + \left(-\frac{\Delta C}{M} + 1\right) x_2(t) + \frac{\Delta K}{M} x_3(t) + \frac{\Delta C}{M} x_4(t) + v_{12}(t) \\ x_3(t+1) = x_3(t) + \Delta x_4(t) + v_{13}(t) \\ x_4(t+1) = x_4(t) + \Delta u(t) + v_{14}(t) \\ y(t) = x_1(t) - x_3(t) + v_2(t) \end{cases}$$

This is normal state-space in discrete time:

$$\mathcal{S}: \begin{cases} x(t+1) = Fx(t) + Gu(t) \\ y(t) = Hx(t) + \overset{0}{\cancel{D}}u(t) \end{cases}$$

$$F = \begin{bmatrix} 1 & \Delta & 0 & 0 \\ -\frac{\Delta K}{M} & -\frac{\Delta C}{M}+1 & \frac{\Delta K}{M} & \frac{\Delta C}{M} \\ 0 & 0 & 1 & \Delta \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad G = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \Delta \end{bmatrix} \quad H = \begin{bmatrix} 1 & 0 & -1 & 0 \end{bmatrix}$$

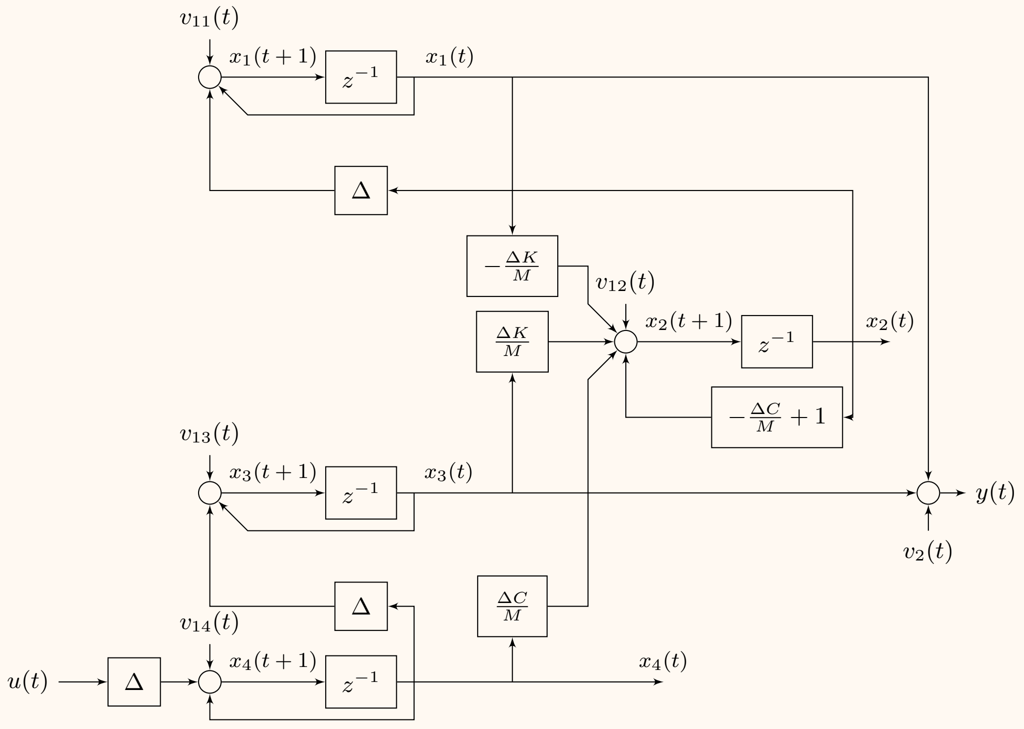
Noise on the state equations:

$$v_1(t) = \begin{bmatrix} v_{11}(t) \\ v_{12}(t) \\ v_{13}(t) \\ v_{14}(t) \end{bmatrix} \sim WN(0, V_1) \qquad V_1 = \begin{bmatrix} \lambda_1^2 & 0 & 0 & 0 \\ 0 & \lambda_1^2 & 0 & 0 \\ 0 & 0 & \lambda_1^2 & 0 \\ 0 & 0 & 0 & \lambda_1^2 \end{bmatrix} \qquad v_2(t) \sim WN(0, V_2)$$

Assumptions:

- All white-noises
- All uncorrelated

V_2 can be estimated by datasheet of elongation sensor, λ_4 can be estimated by datasheet of accelerometer. We expect λ_1, λ_2 and λ_3 to be small, and we can use the simplifying assumption $\lambda_1^2 = \lambda_2^2 = \lambda_3^2 = \lambda^2$, estimated empirically.



We can compute

$$O = \begin{bmatrix} H \\ HF \\ HF^2 \\ HF^3 \end{bmatrix} \qquad R = \begin{bmatrix} G & FG & F^2G & F^3G \end{bmatrix}$$

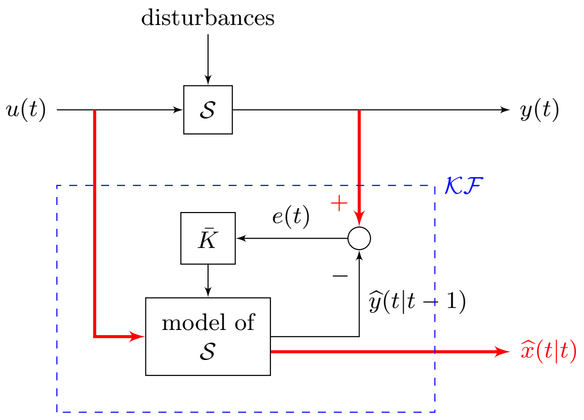
From visual inspection of the block schema we expect full observability from output and full controllability from input.

KF at this point can be applied. Theorem 3.5 and 3.6 are applicable, thus we can directly jump to ARE solution.

Chapter 4

Software-sensing with Black-Box Methods

In chapter 3 we have seen classical technology of software-sensing based on Kalman Filter:



Note If the KF is the asymptotic one (i.e. $K(t) = \bar{K}$), it is a MIMO LTI system.

Main features of this approach:

- A (White-Box/physical) model is needed.
- No need (in principle) of a training dataset including measurements of the state to be estimated.
- It is a feedback estimation algorithm (feedback correction of the model using estimated output error).
- Constructive method (non-parametric, no optimization involved).
- Can be used (in principle) to estimate states which are impossible to be measured (also at prototyping/training/design stage).

Are there other classes of software-sensing techniques? Yes, black-box approaches with *learning/training* from data (system identification).

In this chapter we see them focusing on the architecture (we do not need new algorithms, just use something we have already studied). We will re-cast the SW-sensing problem into a system identification problem.

4.1 Linear Time Invariant Systems

To find the relationship between $u(t) \to \widehat{x}(t|t)$ and $y(t) \to \widehat{x}(t|t)$ we can use a dataset. Indeed, if we have a dataset

$$\{u(1), u(2), \dots, u(N)\}$$
$$\{y(1), y(2), \dots, y(N)\}$$
$$\{x(1), x(2), \dots, x(N)\}$$

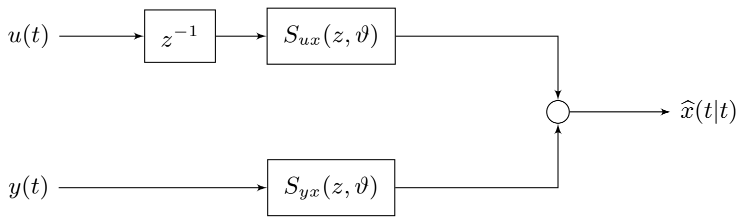
we can estimate the relationship (i.e. TFs) between the inputs ($u(t)$ and $y(t)$) and the output ($x(t)$) without modelling the system and adopting a BB approach instead.

Note This is a supervised training approach, thus, only for the training phase, we need measurements of the state to be estimated (using physical sensor that will be replaced by the trained SW-sensor).

Model selection We focus on this family of models:

$$\widehat{x}(t|t) = S_{ux}(z, \vartheta)u(t-1) + S_{yx}(z, \vartheta)y(t)$$

where we have at least 1-step delay on $u(t)$.



Performance index We define the usual performance index as the *sample variance of the estimation error*:

$$J_N(\vartheta) = \frac{1}{N}\sum_{t=1}^{N} \left(x(t) - (S_{ux}(z, \vartheta)u(t-1) + S_{yx}(z, \vartheta)y(t))\right)^2$$

Optimization

$$\widehat{\vartheta}_N = \arg\min_{\vartheta} J_N(\vartheta)$$

We obtain the Black-Box software sensor $\widehat{x}(t|t)$ as:

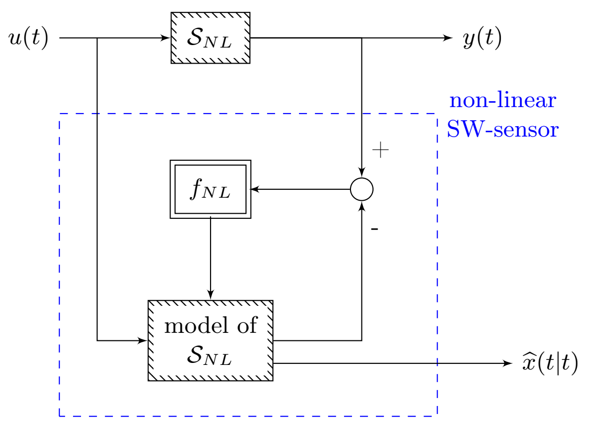
$$\widehat{x}(t|t) = S_{ux}(z, \widehat{\vartheta}_N)u(t-1) + S_{yx}(z, \widehat{\vartheta}_N)y(t)$$

Note Once the SW-sensor has been designed (trained), we no longer need $x(t)$ samples.

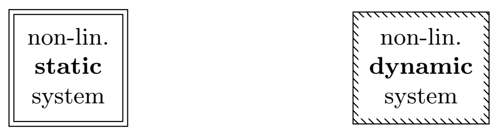
Note The above method is a classic BB parametric approach (using TFs) but the same can also be done using 4-SID algorithm.

4.2 Non-linear Systems

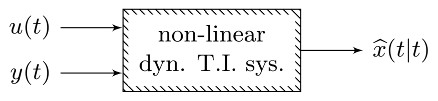
In case the system is non-linear, we can use the same idea used in the LTI case (see 4.1), where we replace the asymptotic KF with a non-linear SW-sensor (like EKF, described in 3.4.2)



Remark — Block scheme notation for non-linear system. Notation used from now on to represent non-linear systems:



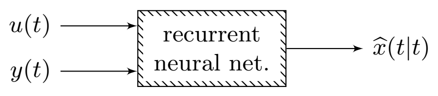
The content of the box (non-linear SW-sensor) is:



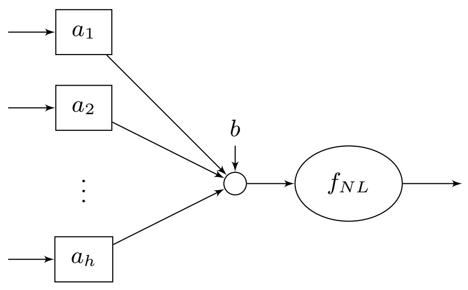
The problem is again the BB identification of a non-linear dynamic system, starting from a measured training dataset.

There are 4 (3+1) different architectures to design the non-linear SW-sensor.

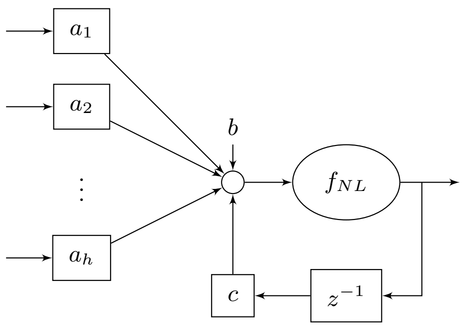
Architecture #1 Use a *Dynamical Recurrent Neural Network (Dynamic RNN)* in which we update *static neurons* into *dynamic neurons*.



If we zoom into a single neuron:



Static neuron (non-linear static system)

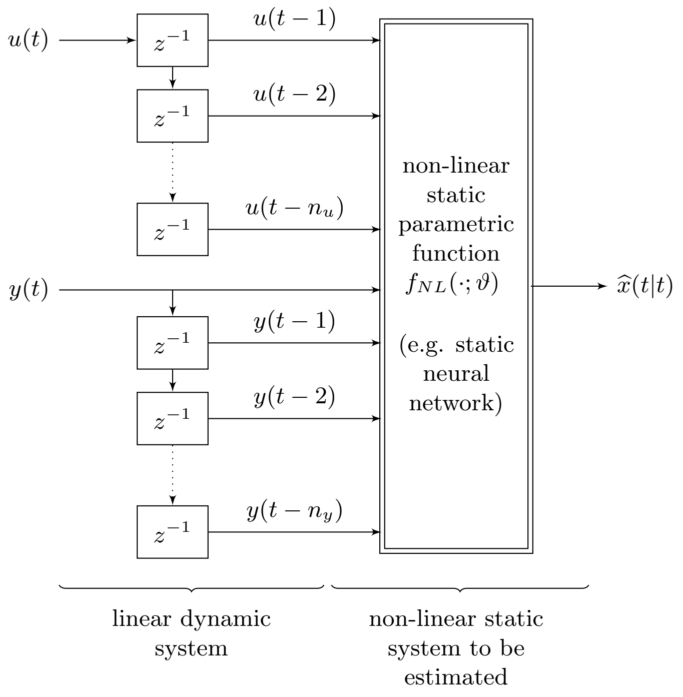


Dynamic neuron (non-linear dynamic system)

where f_{NL} is a non-linear function (e.g. sigmoid function).

Using an RNN with dynamic neurons is the most general approach but it is also practically seldom used due to stability issues and convergence of training.

Architecture #2 Split the SW-sensor into a static non-linear system and a dynamic linear system (namely, a non-recursive FIR scheme)


$u(t)$
z^{-1}
$u(t-1)$
$u(t-2)$
$u(t-n_u)$
$y(t)$
$y(t-1)$
$y(t-2)$
$y(t-n_y)$
non-linear static parametric function $f_{NL}(\cdot;\vartheta)$
(e.g. static neural network)
$\widehat{x}(t|t)$
linear dynamic system
non-linear static system to be estimated

Remark — Pros. Some pros about this architecture:

- Training (supervised) done only to the non-linear static part of the SW-sensor (much simpler than the estimation of an RNN).

- Stability is guaranteed by construction since it is a static FIR architecture.

Remark — **Cons.** In case of a MIMO system with

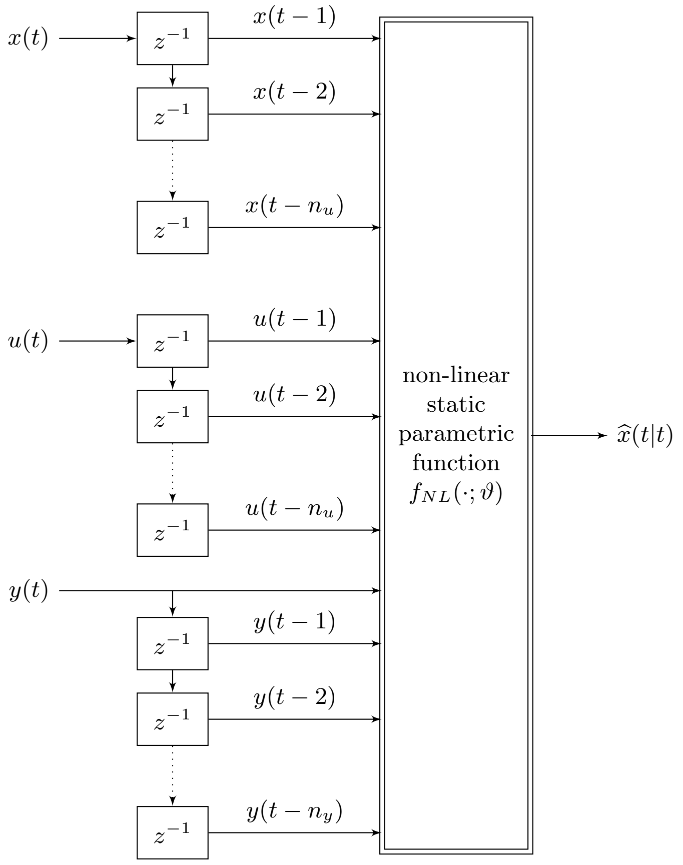
$$m \text{ inputs: } u(t) = \begin{bmatrix} u_1(t) \\ \vdots \\ u_m(t) \end{bmatrix} \quad p \text{ outputs: } y(t) = \begin{bmatrix} y_1(t) \\ \vdots \\ y_p(t) \end{bmatrix} \quad n \text{ states: } x(t) = \begin{bmatrix} x_1(t) \\ \vdots \\ x_n(t) \end{bmatrix}$$

the estimation problem is the search of the optimal parameter vector ϑ for the function

$$f(\cdot;\vartheta) : \mathbb{R}^{m\times n_u + p\times(n_y+1)} \to \mathbb{R}^n$$

Therefore, the I/O size of this non-linear parametric function can be very large.

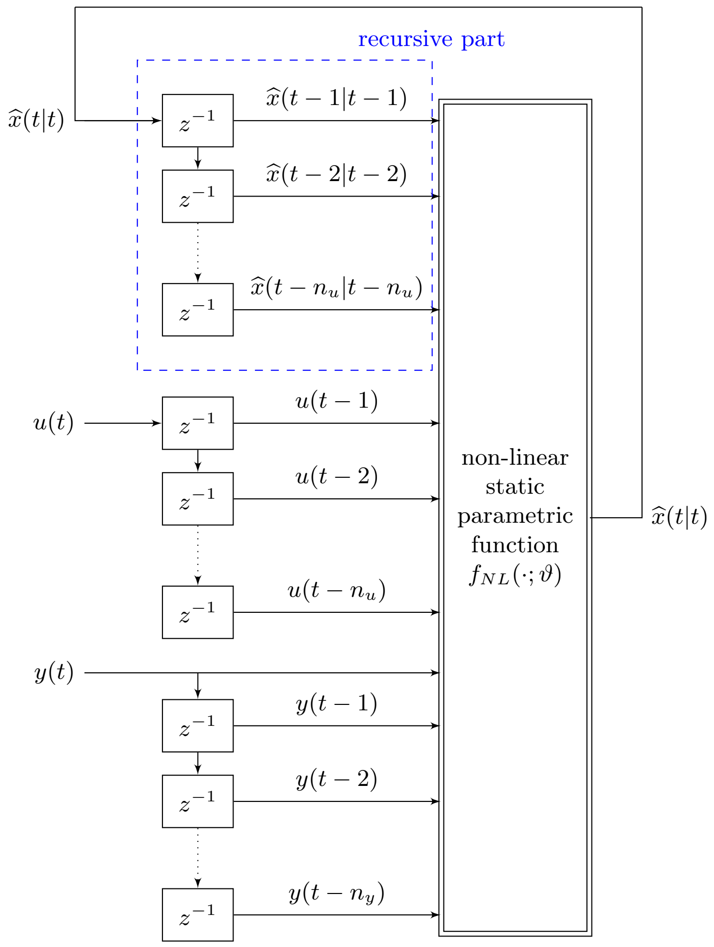
Architecture #3 Like architecture #2, we split the SW-sensor into a static non-linear system and a linear dynamic system but, this time, with a recursive IIR scheme.



Architecture #3 during training phase

Remark — Pros & Cons. The advantage is that usually in IIR architecture n_u and n_y are much smaller (thanks to recursion): lower computation effort.

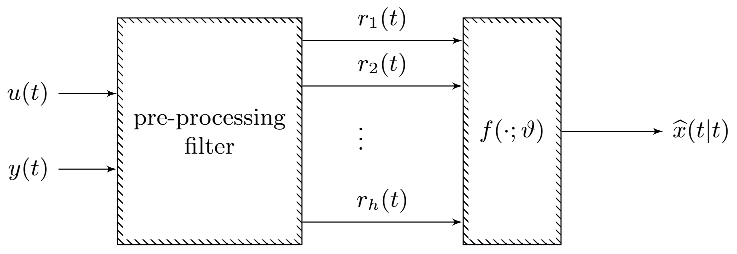
For the disadvantages we have to notice that only for the training part we use $x(t)$ data from a physical sensor; then, in production, the recursion comes to play since we need to plug-in the output $\widehat{x}(t|t)$ in the recursive part: this feedback can provide instability.



Architecture #3 during production phase

Architecture #4 Modification with a-priori signal processing of architectures #1, #2 and #3. The idea is to split the SW-sensor in two stages:

- first stage: from $u(t)$ and $y(t)$, h *regressors* $r_i(t)$ are produced starting from physical knowledge of the system (where h is much smaller than the number of $u(t)$ and $y(t)$ signals)
- second stage: SW-sensor (could be both linear or non-linear and both static or dynamic system) to be firstly trained and then used in production.



The idea is to facilitate the estimation of $f(\cdot;\vartheta)$ by presenting at its input a smaller and more meaningful set of signals (regressors). In this way the BB model identification is much simpler.

Conclusions In case of BB SW-sensing with non-linear systems the problem can be quite complex. Using *brute-force* approach (1 dynamic neural network) is usually doomed to failure. The best is to gain some insight into the system and build some *smart* regressors before black-box map.

4.3 Comparison between KF and BB software sensing

	KF	**BB**
Need of (WB) physical model of the system	Yes	No
Need of a training dataset	No [a]	Yes
Interpretability of the obtained SW-sensor	Yes	No
Easy retuning for a similar (different) system	Yes	No
Accuracy of the obtained SW-sensor	Good	Very Good
Can be used also in case of un-measurable states	Yes	No

[a]In practice some tuning through data is needed.

EXAMPLE — **Example 3.4.2 continued.** Model (key equation) of the system:

$$M\ddot{z} = -c(t)(\dot{z} - \dot{z}_d) - K(z - z_d)$$

Measurable input $\ddot{z}$ with an accelerometer, $z - z_d$ measurable output with elongation sensor. We want to estimate $\dot{z}$.

The change is $c(t)$ is a semi-active suspension, can be electronically changed (control variable).

We can solve the problem with K.F. or we can make an experiment and collect training data:

$$c(t) : \{c(1), c(2), \cdots, c(N)\}$$
$$z(t) - z_d(t) : \{z(1) - z_d(1), z(2) - z_d(2), \cdots, z(N) - z_d(N)\}$$

$$\ddot{z}(t) : \{\ddot{z}(1), \ddot{z}(2), \cdots, \ddot{z}(N)\}$$
$$\dot{z}(t) : \{\dot{z}(1), \dot{z}(2), \cdots, \dot{z}(N)\} \text{ (just for training)}$$

Back to the main equation:

$$M\ddot{z} = -K(z - z_d) - C(t)(\dot{z} - \dot{z}_d)$$

$$\underbrace{\dot{z}}_{\text{to be estim.}} = -\frac{K}{M}\underbrace{\int (z - z_d)dt}_{r_1(t)} - \frac{1}{M}\underbrace{\int C(t)(\dot{z} - \dot{z}_d)dt}_{r_2(t)}$$

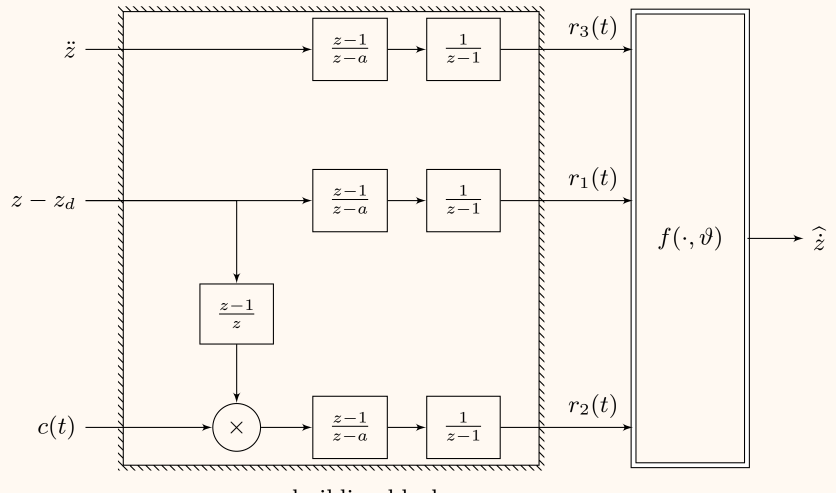
We also consider this equation

$$\dot{z}_d = \underbrace{\int \ddot{z}_d dt}_{r_3(t)}$$

$r_1(t)$ and $r_2(t)$ are the primary regressors, directly linked to $\dot{z}(t)$. $r_3(t)$ is a secondary regressor, it can help $r_1(t)$.

Since these regressors are obtained by integration to avoid drifting (by DC components of noise integration) we have to high-pass the inputs with high-pass filters $\left(\frac{z-1}{z-a}\right)$.

Full filtering scheme



regressors building block

Chapter 5

Gray-Box System Identification

Gray-Box (GB) approach is something between White-Box (where we need the physical model, like KF) and Black-Box (machine learning, data-driven approach).

We'll see two GB approaches:

- using KF theory
- using classical parametric framework

5.1 GB system identification using Kalman Filter

Kalman Filter is not a system identification method, it is a variable estimation approach (software-sensor, observer). However we can use it also for gray-box system identification (*side benefit* of KF).

Problem definition We have a model, typically built as a KF model using first principles (mathematical equations are known):

$$\mathcal{S}: \begin{cases} x(t+1) = f(x(t), u(t); \vartheta) + v_1(t) \\ y(t) = h(x(t); \vartheta) + v_2(t) \end{cases}$$

f and h are linear or non-linear functions, depending on some unknown parameters ϑ (with a physical meaning, e.g. mass, resistance, friction, ...).

The problem is to estimate $\widehat{\vartheta}$ from a dataset.

Problem solution KF solves this problem by transforming the unknown parameters in extended states: KF makes the simultaneous estimation of $\widehat{x}(t|t)$ (classic KF problem) and $\widehat{\vartheta}(t)$ (parameter identification problem).

Trick State extension

$$\mathcal{S}: \begin{cases} x(t+1) = f(x(t), u(t); \vartheta(t)) + v_1(t) \\ \vartheta(t+1) = \vartheta(t) + v_\vartheta(t) \\ y(t) = h(x(t), \vartheta(t)) + v_2(t) \end{cases}$$

The new extended state vector is $x_E = \begin{bmatrix} x(t) \\ \vartheta(t) \end{bmatrix}$. The unknown parameters are transformed in unknown variables.

Remark — New state equation for ϑ estimation. The new equation we created

$$\vartheta(t+1) = \vartheta(t) + v_\vartheta(t)$$

is a *fictitious* equation (not a physical equation).

The core dynamics is $\vartheta(t+1) = \vartheta(t)$, it's the equations of something which is constant. This is exactly the nature of $\vartheta(t)$ which is indeed a constant vector of parameters.

We need a *fictitious* noise in order to force the Kalman Filter to find the right value of ϑ (otherwise KF probably would stay fixed on the initial condition). We're telling KF to do not rely on initial conditions.

Notice that this equation is not of an asymptotic stable system but a simply-stable system. It's not a problem because KF can deal with non-asymptotically stable systems.

Design choice The critical point is, as for any KF problem, the choice of the covariance matrix of $v_\vartheta(t) \sim WN(0, V_\vartheta)$.

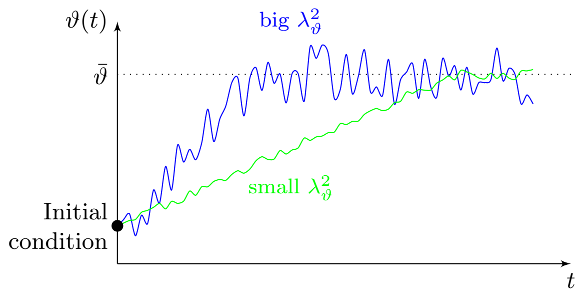
We make the empirical assumption that $v_1 \perp v_\vartheta$ and $v_2 \perp v_\vartheta$ (there is no reason for v_ϑ to be correlated with v_1 and v_2). We also usually make the following strongly simplifying assumption on V_ϑ:

$$V_\vartheta = \begin{bmatrix} \lambda_{1\vartheta}^2 & & & \\ & \lambda_{2\vartheta}^2 & & \\ & & \ddots & \\ & & & \lambda_{n_\vartheta \vartheta}^2 \end{bmatrix}$$

where $\lambda_{1\vartheta}^2 = \lambda_{2\vartheta}^2 = \cdots = \lambda_{n_\vartheta\vartheta}^2 = \lambda_\vartheta^2$. In practice this means that $v_\vartheta(t)$ is a vector of independent WNs all with the same variance λ_ϑ^2. The choice of this single parameter is made empirically: it is a design parameter.

By calling $\bar{\vartheta}$ the true value of ϑ and starting from an initial condition $\vartheta(0)$ (i.e. the best guess on the parameters' value), we can graphically represent the behavior of $\vartheta(t)$ depending on different choice of λ_ϑ^2 (assuming that $\vartheta(t)$ is a single parameter). In particular, we consider the following extreme situations:

- big λ_ϑ^2
- small λ_ϑ^2



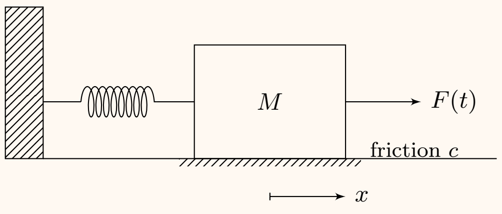
With a small value of λ^2_ϑ there is a slow convergence with small oscillations (noise) at steady-state (big trust to initial conditions). With large values of λ^2_ϑ, instead, there's fast convergence but noisy at steady-state.

λ^2_ϑ is selected according to the best compromise for your specific application.

Note This method can be useful when ϑ is varying (e.g. aging of a resistance).

Note This trick can work in principle with any number of unknown parameters (e.g. 3 sensors, 10 states and 20 parameters). In practice it works well only on a limited number of parameters (e.g. 3 sensors, 5 states and 2 parameters).

Example. Mass linked to the wall with a spring (stiffness K) and with a pulling (or pushing) force $F(t)$.



Input $F(t)$

Output Position $x(t)$ (measured with a physical sensor)

Parameters Realistic case: K and M are known (measured), but the friction c is unknown

$$F(t) \longrightarrow \boxed{\begin{array}{c}\text{system}\\ c=?\end{array}} \longrightarrow x(t)$$

Problem Estimate c with a KF

Since we are using a KF, we do not need a training dataset.

Step 1 Model the system

$$\ddot{x}M = -Kx - c\dot{x} + F(t)$$

It's a differential, continuous time linear equation. It's second order so we need 2 state variables: $x_1(t) = x(t)$ and $x_2(t) = \dot{x}(t)$.

$$\begin{cases} \dot{x}(t) = x_2(t) \\ M\dot{x}_2(t) = -Kx_1(t) - cx_2(t) + F(t) \\ y(t) = x_1(t) \end{cases} \qquad \begin{cases} \dot{x}_1(t) = x_2(t) \\ \dot{x}_2(t) = -\frac{K}{M}x_1(t) - \frac{c}{M}x_2(t) + \frac{1}{M}F(t) \\ y(t) = x_1(t) \end{cases}$$

Step 2 Discretization

Eulero forward (see A): $\dot{x}(t) \approx \frac{x(t+1)-x(t)}{\Delta}$.

$$\begin{cases} \frac{x_1(t+1)-x_1(t)}{\Delta} & = x_2(t) \\ \frac{x_2(t+1)-x_2(t)}{\Delta} & = -\frac{K}{M}x_1(t) - \frac{c}{M}x_2(t) + \frac{1}{M}F(t) \\ y(t) & = x_1(t) \end{cases}$$

$$\begin{cases} x_1(t+1) = x_1(t) + \Delta x_2(t) \\ x_2(t+1) = -\frac{K\Delta}{M}x_1(t) + \left(1 - \frac{c\Delta}{M}\right) x_2(t) + \frac{\Delta}{M}F(t) \\ y(t) = x_1(t) \end{cases}$$

Step 3 State extension: transforming c into a state variable and adding noises.

$$x_3(t+1) = x_3(t) = c(t)$$

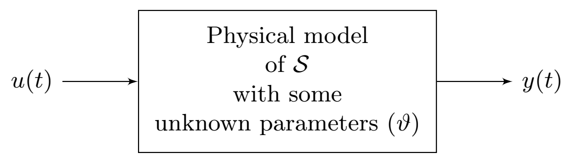
$$\begin{cases} x_1(t+1) = x_1(t) + \Delta x_2(t) + v_{11}(t) \\ x_2(t+1) = -\frac{K\Delta}{M}x_1(t) + \left(1 - \frac{x_3(t)\Delta}{M}\right) x_2(t) + \frac{\Delta}{M}F(t) + v_{12}(t) \\ x_3(t+1) = x_3(t) + v_{13}(t) \\ y(t) = x_1(t) + v_2(t) \end{cases}$$

The system is ready for KF application: we get at the same time $\widehat{x}(t)$ and $\widehat{c}(t)$.

Note We need the Extended Kalman Filter: even if the original system was linear, state extension moved it to a non-linear system (due to the multiplication of $x_3(t)$ and $x_2(t)$ in the second state equation). This is something that happens most of the times when doing the state extension.

5.2 GB system identification using Simulation Error Method

Are there alternative ways to solve Gray-Box system identification problems? A commonly (and intuitive) used method is parametric identification approach based on *Simulation Error Method (SEM)*.


$u(t)$
Physical model of $\mathcal{S}$ with some unknown parameters (ϑ)
$y(t)$

Following the classical 4 steps of a supervised parametric method for system identification:

Step 1 Collect data from an experiment

$$\{\widetilde{u}(1), \widetilde{u}(2), \dots, \widetilde{u}(N)\}$$
$$\{\widetilde{y}(1), \widetilde{y}(2), \dots, \widetilde{y}(N)\}$$

Step 2 Define model structure

$$y(t) = \mathcal{M}(u(t); \bar{\vartheta}, \vartheta)$$

Mathematical model (linear or non-linear) usually written from first principle equations. $\bar{\vartheta}$ is the set of known parameters (mass, resistance, ...), ϑ is the set of unknown parameters (possibly with bounds).

Step 3 Performance index definition (based on *simulation error*)

$$J_N(\vartheta) = \frac{1}{N} \sum_{t=1}^{N} \left(\widetilde{y}(t) - \mathcal{M}(\widetilde{u}(t); \bar{\vartheta}, \vartheta)\right)^2$$

where $\widetilde{y}(t)$ is the measured output and $\mathcal{M}(\widetilde{u}(t); \bar{\vartheta}, \vartheta)$ is the simulated output.

Therefore, $J_N(\vartheta)$ is the *sample variance of the simulated output error.*

Step 4 Optimization

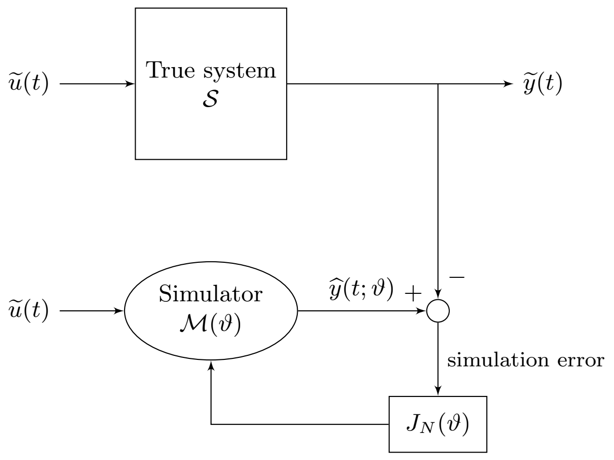
$$\widehat{\vartheta}_N = \arg\min_{\vartheta} J_N(\vartheta)$$

Notes about the optimization step:

- usually no analytic expression of $J_N(\vartheta)$ is available
- each computation of $J_N(\vartheta)$ requires an entire simulation of the model from $t = 1$ to $t = N$
- usually $J_N(\vartheta)$ is a non-quadratic and non-convex function. Iterative and randomized optimization methods must be used
- therefore, it's intuitive but very computationally demanding

5.2.1 Comparison with PEM

We can represent what we've just seen with the following scheme:



In the BB approach, we have seen something very similar but:

- the model was BB
- the performance index was based on *prediction error* (PEM).

The general framework is similar, but with SEM we need the model of the system a-priori.

Example. We collect data $\{\widetilde{u}(1), \widetilde{u}(2), \dots, \widetilde{u}(N)\}$ and $\{\widetilde{y}(1), \widetilde{y}(2), \dots, \widetilde{y}(N)\}$, we want to estimate from data the I/O model.

$$y(t) = \frac{b_0 + b_1 z^{-1}}{1 + a_1 z^{-1} + a_2 z^{-2}} u(t-1) \qquad \vartheta = \begin{bmatrix} a_1 \\ a_2 \\ b_0 \\ b_1 \end{bmatrix}$$

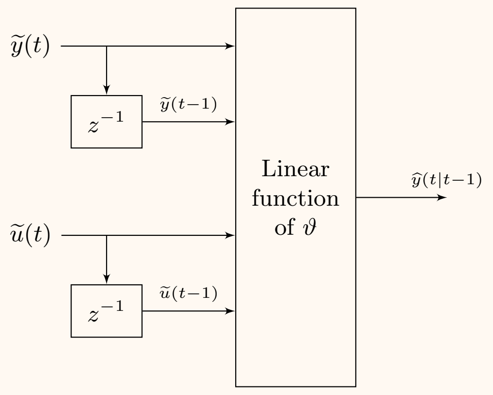
In time domain $y(t) = -a_1 y(t-1) - a_2 y(t-2) + b_0 u(t-1) + b_1 u(t-2)$.

Using PEM

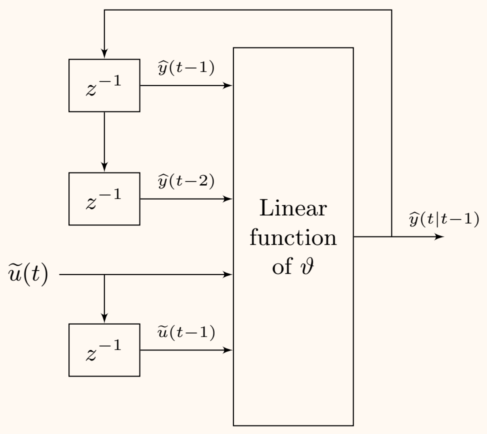
$$\widehat{y}(t|t-1) = -a_1 \widehat{y}(t-1) - a_2 \widehat{y}(t-2) + b_0 \widehat{u}(t-1) + b_1 \widehat{u}(t-2)$$

$$\begin{aligned} J_N(\vartheta) &= \frac{1}{N} \sum_{t=1}^{N} \left(\widetilde{y}(t) - \widehat{y}(t|t-1, \vartheta)\right)^2 \\ &= \frac{1}{N} \sum_{t=1}^{N} \left(\widetilde{y}(t) + a_1 \widetilde{y}(t-1) + a_2 \widetilde{y}(t-2) - b_0 \widetilde{u}(t-1) - b_1 \widetilde{u}(t-2)\right)^2 \end{aligned}$$

Notice that it's a quadratic formula.



PEM



SEM

Using SEM

$$\widehat{y}(t|t-1) = -a_1 \widehat{y}(t-1) - a_2 \widehat{y}(t-2) + b_0 \widetilde{u}(t-1) + b_1 \widetilde{u}(t-2)$$

$$
\begin{aligned}
J_N(\vartheta) &= \frac{1}{N}\sum_{t=1}^{N}\left(\widetilde{y}(t)-\widehat{y}(t|t-1,\vartheta)\right)^2 \\
&= \frac{1}{N}\sum_{t=1}^{N}\left(\widetilde{y}(t)+a_1\widehat{y}(t-1)+a_2\widehat{y}(t-2)-b_0\widetilde{u}(t-1)-b_1\widetilde{u}(t-2)\right)^2
\end{aligned}
$$

Notice that it's non-linear with respect to ϑ.

PEM approach looks much better, but do not forget the noise! PEM is much less robust w.r.t. noise, we must include a model of the noise in the estimated model. We use ARMAX models.

If we use Auto Regressive with Exogenous Input (ARX) models:

$$
\begin{aligned}
y(t) &= \frac{b_0+b_1z^{-1}}{1+a_1z^{-1}+a_2z^{-2}}u(t-1)+\frac{1}{1+a_1z^{-1}+a_2z^{-2}}e(t) \\
\widehat{y}(t|t-1) &= b_0u(t-1)+b_1u(t-2)-a_1y(t-1)-a_2y(t-2)
\end{aligned}
$$

If we use ARMAX models the numerator of the T.F. for $e(t)$ is $1+c_1z^{-1}+\ldots+c_mz^{-m}$, in this case $J_N(\vartheta)$ is non-linear. This leads to the same complexity of SEM

The second problem of PEM is high sensitivity to sampling time choice. Remember that when we write at discrete time $y(t)$ we mean $y(t\cdot\Delta)$.

$$
\widehat{y}(t|t-1) = -a_1\widetilde{y}(t-1)-a_2\widetilde{y}(t-2)+b_0\widetilde{u}(t-1)+b_1\widetilde{u}(t-2)
$$

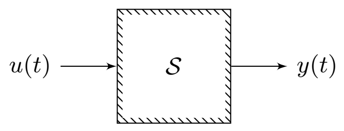
If Δ is very small the difference between $\widetilde{y}(t)$ and $\widetilde{y}(t-1)$ becomes very small. The effect is that the PEM optimization ends to provide this *trivial* solution:

$$
a_1=-1 \qquad a_2\to 0 \qquad b_0\to 0 \qquad b_1\to 0 \qquad \Rightarrow \qquad \widetilde{y}(t)\approx\widetilde{y}(t-1)
$$

This is a wrong model due to the fact that the recursive part of the model is using past measures of the output instead of past values of the model outputs.

5.3 Summary of system identification methods for I/O systems

The problem is to estimate the model of a system $\mathcal{S}$ that, in general, could be non-linear and dynamic.



In order to do that we have to:

- collect a dataset for training (supervised method): can be with or without design of the experiment

- choose a model domain (linear static/non-linear static/linear dynamic/non-linear dynamic), using GB or BB approach
- choose an estimation framework: constructive (4SID), parametric (PEM or SEM) or filtering (state extension of KF)

> **REMARK — Final question.** For system identification and SW-sensing is better Black-Box or White-Box?
>
> It depends on the goals and type of applications:
>
> - BB is very general and very flexible, make maximum use of data and no or little need of domain know-how
> - WB is very useful when you are the system-designer (not only the control algorithm designer): it can provide more insight in the system.
> - GB can sometimes be obtained by hybrid systems (part is black-box and part is white-box).

Chapter 6

Minimum Variance Control

Minimum Variance Control (MVC) is about design and analysis of feedback systems, it is not about system identification nor software sensing.

Why we dedicate a chapter on *control*?

- Control design is the main motivation to system identification and software sensing.
- MVC is based on *mathematics* of system identification and software sensing (prediction theory).
- MVC can be considered as a general tool of stochastic optimization of feedback systems.

6.1 MVC System

Setup of the problem Consider a generic ARMAX model

Recall — ARMAX system.

$$y(t) = \frac{B(z)}{A(z)}u(t-k) + \frac{C(z)}{A(z)}e(t) \qquad e(t) \sim WN(0, \lambda^2)$$

$$\begin{aligned} B(z) &= b_0 + b_1 z^{-1} + \cdots + b_p z^{-p} \\ A(z) &= 1 + a_1 z^{-1} + \cdots + a_m z^{-m} \\ C(z) &= 1 + c_1 z^{-1} + \cdots + c_n z^{-n} \end{aligned}$$

Note: The input is $u(t)$ and the output is $y(t)$. The noise $e(t)$ is a non-measurable input that models the uncertainty of the system.

Assumptions:

- $\frac{C(z)}{A(z)}$ is in *canonical form*
- $b_0 \neq 0$ (this implies that $k \geq 0$ is the actual delay)
- $\frac{B(z)}{A(z)}$ is *minimum phase*

Recall — Transfer function in canonical form. A transfer function $W(z) = \frac{C(z)}{A(z)}$ is in *canonical form* if

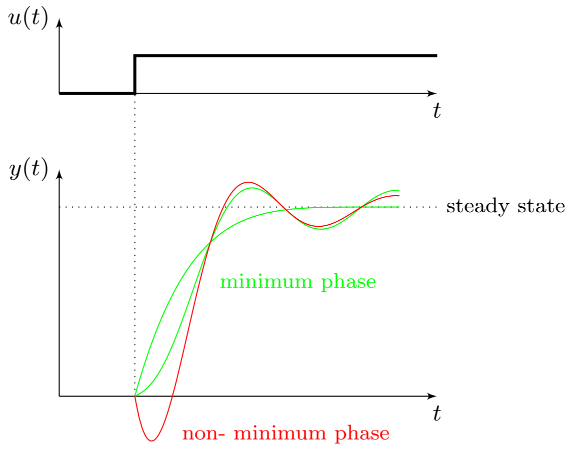
1. $C(z)$ and $A(z)$ are monic (i.e. the coefficients of the maximum degree terms of $C(z)$ and $A(z)$ are equal to 1);
2. $C(z)$ and $A(z)$ have null relative degree;
3. $C(z)$ and $A(z)$ are coprime (i.e. they have no common factors);

4a. the poles of $W(z)$ are such that $|z| < 1$;

4b. the zeros of $W(z)$ are such that $|z| < 1$. (more stringent condition than the one we saw in MIDA1)

Remark. $\frac{B(z)}{A(z)}$ is said to be *minimum phase* if all the roots of $B(z)$ are strictly inside the unit circle.

What does it means in practice?



At the very beginning the response of a non-minimum phase system goes to the opposite direction w.r.t. the final value. Intuitively it's very difficult to control non-minimum phase systems: you can take the wrong decision if you react immediately.

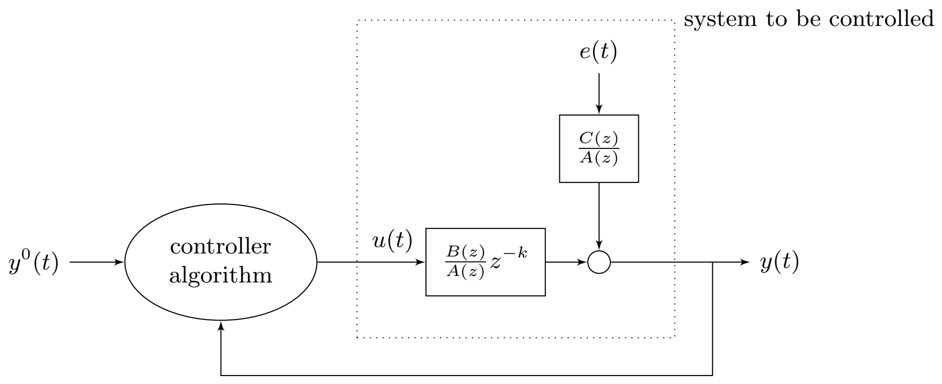
Also for human it's difficult, for example *steer to roll* dynamics in a bicycle: if you want to steer left, you must first steer a little to the right and then turn left.

Design of controller for non-minimum phase is difficult and requires special design techniques (no MVC but *generalized MVC (GMVC)*, described in 6.3.1).

Some additional (small) technical assumptions:

- $y^0(t)$ and $e(t)$ are not correlated (usually fulfilled).
- We assume (worst case) that $y^0(t)$ is known only up to time t (present time): we have no preview of the future desired $y^0(t)$ (i.e. $y^0(t)$ is totally unpredictable or $\widehat{y^0}(t+k|t) = y^0(t)$).

Goal of the problem The problem we wish to solve is optimal tracking of the desired behavior of output (which is the classical goal of control systems):



where $y^0(t)$ os the desired output value, called *reference.*

In a more formal way MVC is an optimization control problem that tries to find $u(t)$ that minimize this performance index:

$$J = E\left[(y(t) - y^0(t))^2\right]$$

where J is the variance of the tracking error: that's why it's called Minimum Variance Control.

Solution of the problem The main trick is to split $y(t)$ into $\widehat{y}(t|t-k)$, the *predictor*, and $\epsilon(t)$, the *prediction error.*

Now we can write J as

$$\begin{aligned} J =& \mathbb{E}\left[\left(\widehat{y}(t|t-k) + \epsilon(t) - y^0(t)\right)^2\right] \\ =& \mathbb{E}\left[\left((\widehat{y}(t|t-k) - y^0(t)) + \epsilon(t)\right)^2\right] \\ =& \mathbb{E}\left[\left((\widehat{y}(t|t-k) - y^0(t)\right)^2\right] + \mathbb{E}[\epsilon(t)^2] + 2\mathbb{E}\left[\epsilon(t)\left(\widehat{y}(t|t-k) + y^0(t)\right)\right] \end{aligned}$$

where we note that

- $\mathbb{E}[\epsilon(t)^2]$ doesn't depend on $u(t)$: it's not subject to minimization, since it's just a constant number.
- $\mathbb{E}[\epsilon(t)y^0(t)] = 0$, since $e(t) = f(e(t), e(t-1), \dots)$ and, by assumption, $y^0 \perp e$.
- $\mathbb{E}[\epsilon(t)\widehat{y}(t|t-k)] = 0$, since by construction $\epsilon(t) \perp \widehat{y}(t|t-k)$ (otherwise $\widehat{y}(t|t-k)$ wouldn't be the optimal predictor).

Therefore, minimizing J is equivalent to minimizing $\mathbb{E}\left[\left((\widehat{y}(t|t-k) - y^0(t)\right)^2\right]$, which is minimized if

$$\widehat{y}(t|t-k) = y^0(t) \tag{6.1}$$

Now we have to compute $\widehat{y}(t|t-k)$ and impose $\widehat{y}(t+k|t) = y^0(t+k)$ (obtained by shifting the equation 6.1 using the z operator).

However, by assumption, at time t we don't know $y^0(t+k)$: the best we can do is to replace it with the last available value of y^0, that is $y^0(t)$.

Therefore, the condition 6.1 becomes

$$\widehat{y}(t+k|t) = y^0(t) \tag{6.2}$$

Recall — k-step predictor of an ARMAX system.

$$\widehat{y}(t+k|t) = \frac{B(z)E(z)}{C(z)}u(t) + \frac{\widetilde{R}(z)}{C(z)}y(t)$$

where $E(z)$ is the *result* and $R(z) = \widetilde{R}(z)z^{-k}$ is the *residual* of the k-step long division between $C(z)$ and $A(z)$.

By plugging in the k-step predictor of an ARMAX system formula in the equation 6.2 we obtain

$$\frac{B(z)E(z)}{C(z)}u(t) + \frac{\widetilde{R}(z)}{C(z)}y(t) = y^0(t)$$

and by making $u(t)$ explicit we obtain the *General Formula of MVC*:

$$u(t) = \frac{1}{B(z)E(z)}\left(C(z)y^0(t) - \widetilde{R}(z)y(t)\right) \tag{6.3}$$

The block scheme representation of the MVC system is

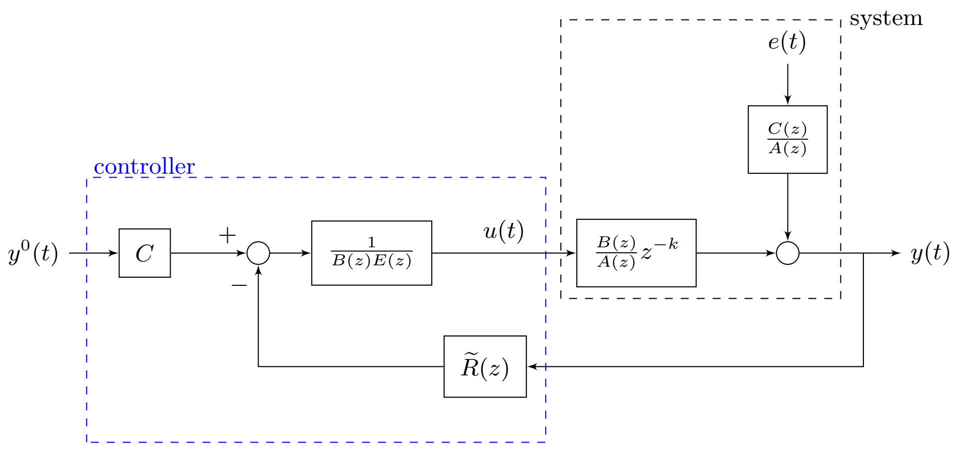
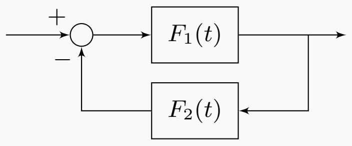


Figure 6.1: MVC System

6.2 Analysis of the MVC System

6.2.1 Stability of the MVC System

Recall. For stability let's recall a result of *negative feedback system*:



To check the closed-loop stability:

- compute the *loop-function* $L(z) = F_1(z)F_2(z)$ (**remember**: do not simplify!)
- build the *characteristic polynomial* $\chi(z) = L_N(z) + L_D(z)$ (sum of the numerator and the denominator of $L(z)$)
- find the roots of $\chi(z)$: closed loop system is asymptotically stable if and only if all the roots of $\chi(z)$ are strictly inside the unit circle

Check the stability of the MVC system represented in figure 6.1

$$L(z) = \frac{1}{B(z)E(z)} \cdot \frac{z^{-k}B(z)}{A(z)} \cdot \widetilde{R}(z)$$

$$\begin{aligned}\chi(z) &= z^{-k}B(z)\widetilde{R}(z) + B(z)E(z)A(z) \\ &= B(z)\underbrace{\left(z^{-k}\widetilde{R}(z) + E(z)A(z)\right)}_{C(z)} \\ &= B(z)C(z)\end{aligned}$$

where we used the following result of the long division between $C(z)$ and $A(z)$:

$$\frac{C(z)}{A(z)} = E(z) + \frac{z^{-k}\widetilde{R}(z)}{A(z)}$$

The MVC system is always guaranteed asymptotically stable, since the roots of $\chi(z)$ are the roots of $B(z)$ and $C(z)$ and:

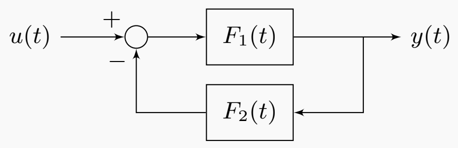
- all roots of $B(z)$ are strictly inside the unit circle (thanks to minimum phase assumption on $\frac{B(z)}{A(z)}$)
- all roots of $C(z)$ are inside the unit circle (thanks to the assumption of canonical representation of $\frac{C(z)}{A(z)}$)

6.2.2 Performance of the MVC System

The system can be rewritten as follows considering the two inputs: $y^0(t)$ and $e(t)$ (non-measurable input)

$$y(t) = F_{y^0y}(z)y^0(t) + F_{ey}(z)e(t)$$

Recall. The transfer function from the input to the output of a *negative feedback system* can be computed with:



$$F(z) = \frac{F_1(z)}{1 + F_1(z)F_2(z)} \qquad y(t) = F(z)u(t)$$

where we recall that $F_1(z)F_2(z)$ is the loop-function $L(z)$ and $F_1(z)$ is the direct line from the input to the output.

Therefore, the TF from $y^0(t)$ to $y(t)$ is

$$\begin{aligned} F_{y^0y}(z) =& \frac{C(z) \cdot \frac{1}{B(z)E(z)} \cdot \frac{z^{-k}B(z)}{A(z)}}{1 + \underbrace{\frac{1}{B(z)E(z)} \cdot \frac{z^{-k}B(z)}{A(z)} \cdot \widetilde{R}(z)}_{L(z)}} \\ =& \dots \\ =& z^{-k} \end{aligned}$$

where we remind that $L(z)$ has been already computed for the stability check (6.2.1).

Similarly, the TF from $e(t)$ to $y(t)$ is

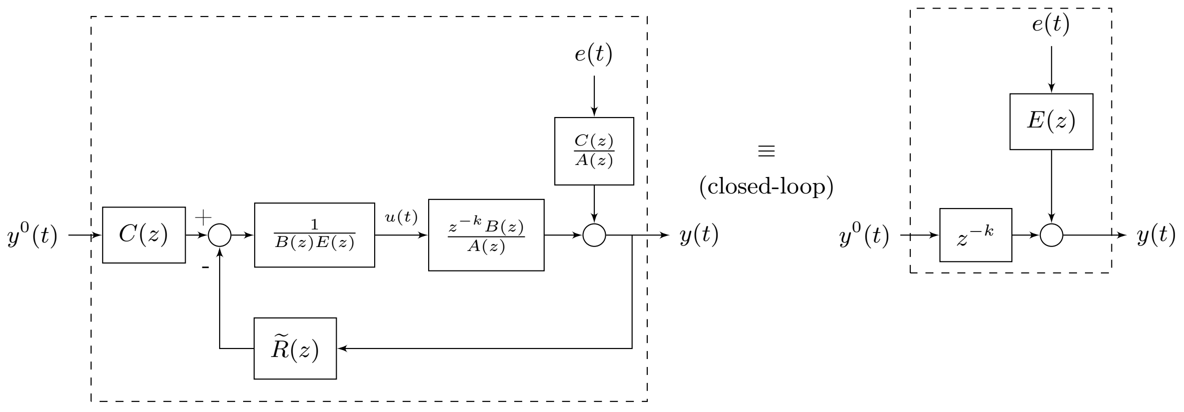
$$\begin{aligned} F_{ey}(z) =& \frac{\frac{C(z)}{A(z)}}{1 + \underbrace{\frac{1}{B(z)E(z)} \cdot \frac{z^{-k}B(z)}{A(z)} \cdot \widetilde{R}(z)}_{L(z)}} \\ =& \dots \\ =& E(z) \end{aligned}$$

Thus we can say that

$$\begin{aligned} y(t) =& F_{y^0y}(z)y^0(t) + F_{ey}(z)e(t) \\ =& z^{-k}y^0(t) + E(z)e(t) \\ =& y^0(t-k) + E(z)e(t) \end{aligned}$$

which is the very simple closed-loop relationship between input and output in a MVC system.

Remark. The closed-loop behavior is very simple. Visually:



In principle the *ideal* desired behavior would be $y(t) = y^0(t) + \cancelto{0}{E(z)}e(t)$, but, since the system has internally k-step delay and the reference $y^0(t)$ is not predictable, the *optimal* result is the one found in 6.2.2, that is:

$$y(t) = y^0(t-k) + E(z)e(t)$$

Note Where are all the poles of the original system?

MVC *pushes* all the system poles into the non-observable and/or non-controllable parts of the system (it makes internal cancellations). However, this is something wanted and therefore it is not a problem since we verified it's internally asymptotically stable.

6.3 Main limitations of MVC

- Can be applied only to minimum-phase systems.
- We cannot moderate the control/actuation effort.
- We cannot design a specific behavior from $y^0(t)$ to $y(t)$.

6.3.1 Generalized Minimum Variance Control (GMVC)

To overcome these limits there is an extension called *Generalized Minimum Variance Control (GMVC).* The main difference w.r.t. MVC is the extension of the performance index:

$$\text{MVC: } J = E\left[\left(y(t) - y^0(t)\right)^2\right]$$

$$\text{GMVC: } J = E\left[\left(P(z)y(t) - y^0(t) + Q(z)u(t)\right)^2\right]$$

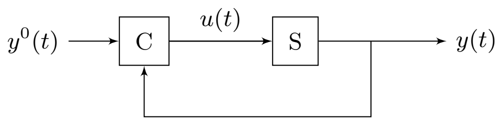
where $P(z)$ and $Q(z)$ are TFs designed by engineers; in particular:

- $P(z)$ is the *reference model*
- $Q(z)u(z)$ is used to penalize the usage of control action, which has to be moderated since it's expensive in terms of energy, power, aging, ...

Observation. The performance index J of the MVC is just a special case of the GMVC's one, where $P(z) = 1$ and $Q(z) = 0$.

Remark — Another important difference w.r.t. MVC. GMVC can be applied also to non-minimum phase systems.

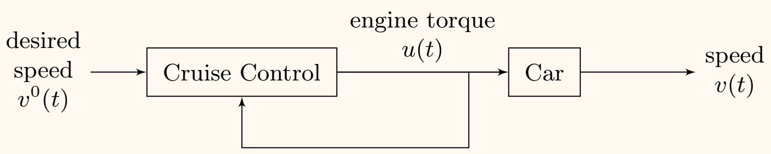
Remark — Reference model $P(t)$. In a generic feedback control system

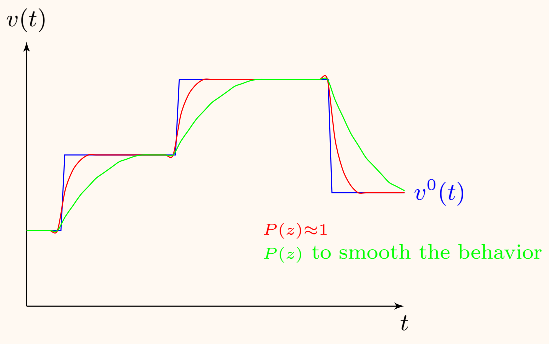


the typical goal is to obtain the best possible tracking $y(t) = y^0(t)$. However, perfect tracking may not be the best solution: sometimes the best solution is to track a *reference model* or *reference behavior* from $y^0(t)$ to $y(t)$:

$$y(t) = P(z)y^0(t)$$

Example — Cruise control in a car. Cruise control is a typical example of *reference model*'s usage.





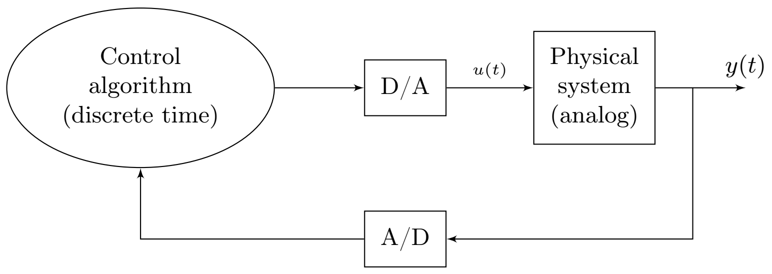
In this scenario it is intuitive that a very aggressive tracking (i.e. $P(z) \approx 1$) is unwanted since it is uncomfortable and dangerous. A smoother reference behavior is preferred (i.e. $P(z)$ is a low-pass filter).

Appendices

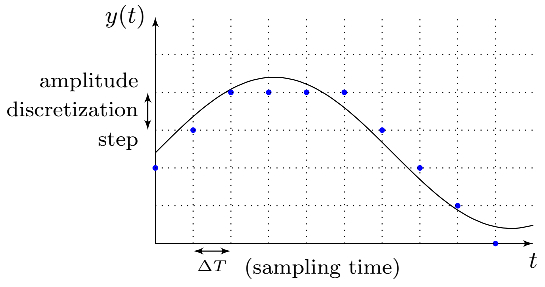
Appendix A

Discretization of Analog Dynamical systems

Discretization allows us to do modeling, SW-sensing and control in a digital context.



Analog to Digital Converter (A/D)



Time discretization ΔT is the sampling time

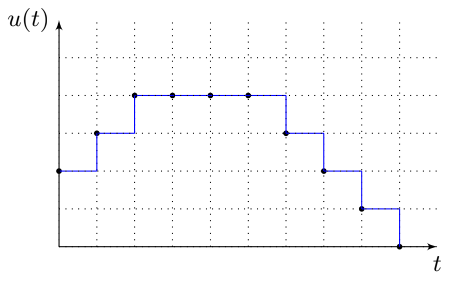
Amplitude discretization step Number of levels used for discretization, e.g. *10-bit discretization* uses 2^{10} levels of amplitude

An high quality A/D converter:

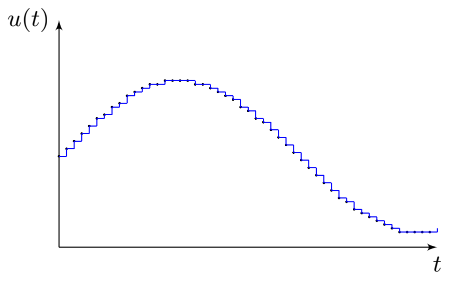
- small ΔT

- high number of levels (e.g. 16-bits)

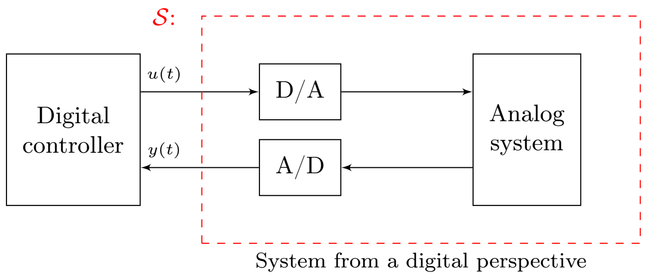
Digital to Analog Converter (D/A)



If ΔT is sufficiently small, the step-wise analog signal is very similar to a smooth analog signal (the steps become "invisible").



Model of the Digital Perspective



Note Both $u(t)$ and $y(t)$ are digital signal.

We can obtain a discrete-time model from $u(t)$ to $y(t)$ for the system $\mathcal{S}$:

$$\begin{tikzcd} u(t) \arrow[r] & \boxed{\mathcal{S}} \arrow[r] & y(t) \end{tikzcd}$$

and we can either:

- make BB system identification starting from measured data: directly estimate a discrete-time model
- starting from a physical WB model (continuous time) that we need to discretize

Let's analyze the latter method.

Discretization of a physical WB model

We introduce two methods for discretize a physical WB model.

1) State-Space Transformation The most used approach is the *State-Space Transformation.*

$$\mathcal{S}: \begin{cases} \dot{x} = Ax + Bu \\ y = Cx + (Du) \\ \qquad (A,\ B,\ C,\ D \text{ cont. time}) \end{cases} \xrightarrow[\Delta T]{\text{discretization}} \mathcal{S}: \begin{cases} x(t+1) = Fx(t) + Gu(t) \\ y(t) = Hu(t) + (Du(t)) \\ \qquad (F,\ G,\ H,\ D \text{ discrete time}) \end{cases}$$

Discretization is done via those transformation formulas:

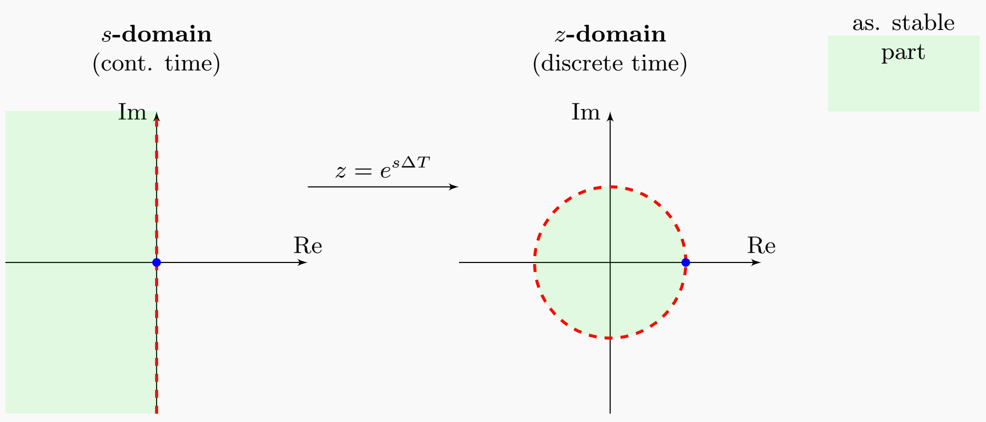
$$\begin{aligned} F &= e^{A\Delta T} \\ G &= \int_0^{\Delta T} e^{A\delta} B\, d\delta \\ H &= C \\ D &= D \end{aligned}$$

Recall — ***s*-domain.** Let's recall the transformation from the s-domain to the z-domain.

Poles How the poles of the continuous time system are transformed?

It can be proved that the eigenvalues (poles) follow the *sampling transformation rule.*

$$z = e^{s\Delta T} \qquad \lambda_F = e^{\lambda_A \Delta T}$$



Zeroes How the zeroes of the continuous time system are transformed?

Unfortunately there is no simple rule like the poles. We can only say:

$$G(s) = \frac{\text{polynomial in } s \text{ with } h \text{ zeros}}{\text{polynomial in } s \text{ with } k \text{ poles}} \qquad \text{if } G(s) \text{ is strictly proper: (i.e. } k > h)$$

$$G(z) = \frac{\text{polynomial in } z \text{ with } k-1 \text{ zeros}}{\text{polynomial in } z \text{ with } k \text{ poles}} \qquad G(z) \text{ with relative degree 1}$$

We have new $k - h - 1$ zeros that are generated by the discretization. They are called *hidden zeros.*

Unfortunately these hidden zeros are frequently outside the unit circle, which means that $G(z)$ is not minimum phase even if $G(s)$ is minimum phase.

Therefore, we need for instance GMVC to design the control system.

2) Time-derivative discretization Another simple discretization technique frequently used is the discretization of time-derivative $\dot{x}$.

$$\textbf{Eulero backward} \qquad \dot{x} \approx \frac{x(t) - x(t-1)}{\Delta T} = \frac{x(t) - z^{-1}x(t)}{\Delta T} = \frac{z-1}{z\Delta T}x(t)$$

$$\textbf{Eulero forward} \qquad \dot{x} \approx \frac{x(t+1) - x(t)}{\Delta T} = \frac{zx(t) - x(t)}{\Delta T} = \frac{z-1}{\Delta T}x(t)$$

General formula

$$\dot{x}(t) \approx \left[\frac{z-1}{\Delta T}\frac{1}{\alpha z + (1-\alpha)}\right]x(t) \qquad \text{with } 0 \leq \alpha \leq 1$$

We consider three special cases:

- if $\alpha = 0$ it's Eulero Forward
- if $\alpha = 1$ it's Eulero Backward
- if $\alpha = \frac{1}{2}$ it's Tustin method (mostly used)

Choice of the sampling time

The critical choice is ΔT (sampling time).

Simple idea The general intuitive rule is: the smaller ΔT, the better.

RECALL — **Sampling and Nyquist frequency.** If ΔT is sampling time, then

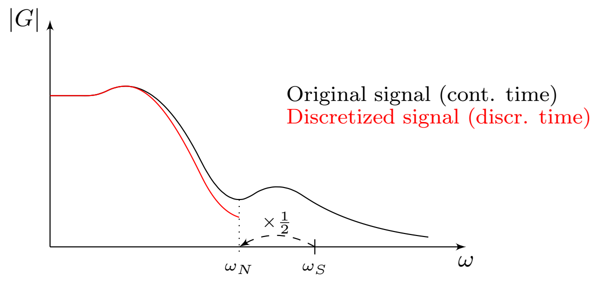
Notation

$$f_S = \frac{1}{\Delta T}[\text{Hz}] \qquad \omega_S = \frac{2\pi}{\Delta T}[\text{rad/s}] \qquad f_N = \frac{1}{2}f_S[\text{Hz}] \qquad \omega_N = \frac{1}{2}\omega_S[\text{rad/s}]$$

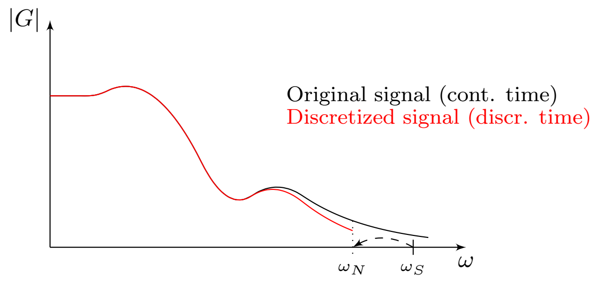
where f_S is the *sampling frequency* and f_N is the *Nyquist frequency.*

Spectrum of a discretized signal For a discretized signal its spectrum is limited and it ranges over $[0, \omega_N]$; as a result, the fitting between the spectrum of the discretized signal and the one of the original signal is very close at low frequencies and becomes more inaccurate as we approach ω_N.

If ΔT is large, f_S is small (and accordingly, f_N too).



Otherwise, if ΔT is smaller, ω_S (and ω_N accordingly) larger.



As we can see from the two graphs, in both case we have approximations of the original spectrum but in the latter case this approximation is valid over a larger *bandwidth*.

Drawbacks Hidden problems of a too-small ΔT:

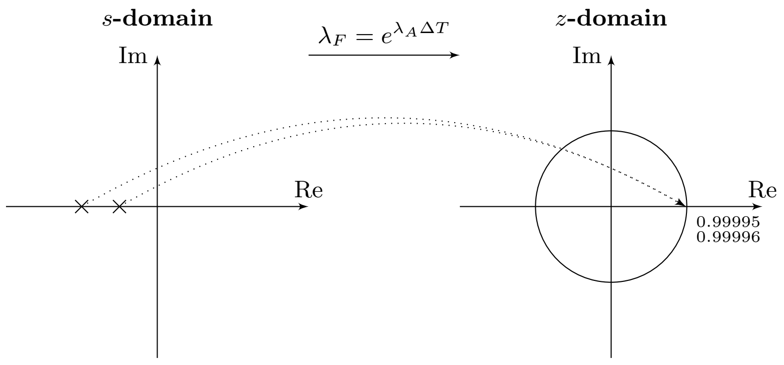
- sampling devices (A/D and D/A) cost
- computational cost (e.g. update an algorithm every $1\mu s$ is much heavier than every $1ms$)
- cost of memory (if data storing is needed)
- numerical precision cost

The latter one is the most critical one and it is really an hidden problem.

Numerical precision cost Let's make a numerical example to understand it.

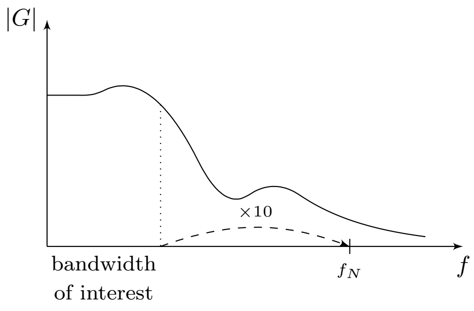
Consider a continous time TF with two asymptotically stable poles (eigenvalues) in $s = -3$ and $s = -2$. These can be mapped in the z-domain using the formula $\lambda_F = e^{\lambda_A \Delta T}$. If ΔT is very small (tends to zero), we squeeze all the poles very closed to $(1, 0)$.

Graphically:



Therefore, we need very high numerical precision (use a lot of digits) to avoid instability.

Rule of thumb of control engineers: f_S is between 10 and 20 times the system bandwidth we are interested in.

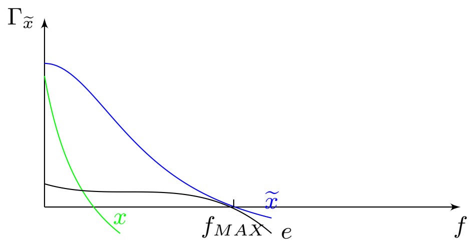


Usually, the bandwidth of interest is the one of the closed-loop control system.

Aliasing problem Another problem is managing the *aliasing* problem, which is a big and critical problem in the A/D step.

Theorem A.1 — Nyquist–Shannon sampling theorem. The maximum frequency content of a signal f_{MAX} to be sampled should be such that $f_{MAX} \leq f_N$.

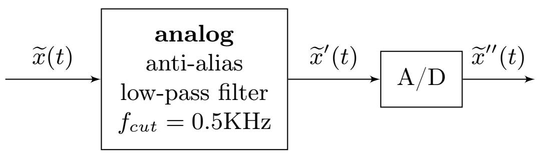
When we want measure a signal x we capture also the measurement noise; indeed, what we really obtain is $\widetilde{x}(t) = x(t) + e(t)$, where e is the noise. Hence, the spectrum will also be composed by the spectrum of the original signal x and the spectrum of the noise.



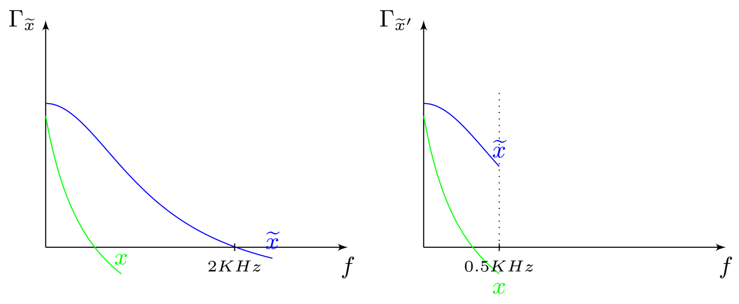
Therefore, if we want to sample the measured signal, and for example, $f_{MAX} = 2$KHz, then we need $f_N \geq 2\text{KHz} \implies f_S \geq 4\text{KHz}$. On the other hand, we know that the bandwidth of the original signal $x(t)$ will be much smaller due to the presence of the noise. For example, suppose that the frequency content of $x(t)$ is contained in the range $[0, 0.5]$ KHz: we can therefore sample with an A/D that samples at $f_S = 1$ KHz.

Analog solution The classical way to deal with aliasing is to use anti-alias analog filters.

> **Recall — Anti-Aliasing Filter (AAF).** An *anti-aliasing filter (AAF)* is a filter used before a signal sampler to restrict the bandwidth of a signal to satisfy the Nyquist–Shannon sampling theorem (A.1) over the band of interest.
>
> In practice, it is an analog low-pass filter with a cut frequency $f_{cut} \leq \frac{1}{2} f_S = f_N$.

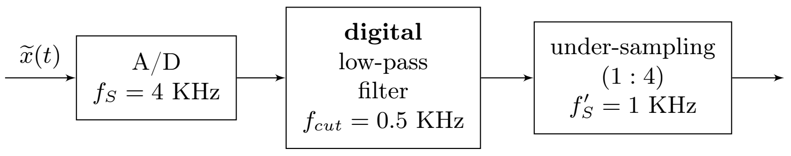


In this case, since $f_S = 1$ KHz we need a low-pass filter that cut everything above $f_N = \frac{1}{2} f_S = 0.5$ KHz.



In this way the signal $\widetilde{x}$ is ready to be sampled: aliasing is avoided ($f'_{MAX} = f_{cut} = f_N$); furthermore, all the frequency components in the bandwidth of the original signal $x(t)$ are preserved and the ones introduced by the noise are cut-off.

Full digital solution Today the most used solution is the full digital one in which there is no more the need of an analog anti-alias filter.



With this approach we consider the bandwidth of the noisy signal $\widetilde{x}(t)$ ($f_{MAX} = 2$ KHz) and we sample it at $f_S = 2 f_{MAX} = 4$ KHz. Then we cut-off all the frequency components introduced by the noise with a digital low-pass filter ($f_{MAX} \rightarrow f'_{MAX} = 0.5$ KHz). Last, since now the bandwidth of the signal is $f'_{MAX} = 0.5$ KHz, we can re-sample the signal at $f'_S = 2f'_{MAX} = 1$ KHz: this is *under-sampling (1:4)* (because $f'_S = \frac{1}{4}f_S$) which in practice means to take one sample every out of four.

Acronyms

AR Auto Regressive.
ARE Algebraic Riccati Equation.
ARMA Auto Regressive Moving Average.
ARMAX Auto Regressive Moving Average with Exogenous Input.
ARX Auto Regressive with Exogenous Input.

BB Black-Box.

DRE Difference Riccati Equation.

GB Gray-Box.

IR Impulse Response.

KF Kalman Filter.

MA Moving Average.
MSE Mean Square Error.

PEM Prediction Error Minimization.

SEM Simulation Error Method.
SP Stochastic Process.
SS State-Space.
SSP Stationary Stochastic Process.

TF Transfer Function.

WB White-Box.
WN White Noise.