Politicnico di Milano
051589 – MIDA2

# Model Identification and Data Analysis<br>Part 2

Instructors

Prof. Sergio Savaresi
Ing. Stefano Dattilo

Author

Edoardo Morassutto

Contributors

Marco Donadoni
Cosimo Russo
Federico Cazzola

A.Y. 2019 – 2020

# Contents

0 Introduction 4

0.1 General topics of MIDA course . . . 4

0.1.1 Super summary of MIDA 1 . . . 4

0.1.2 MIDA 2 . . . 5

0.2 Motivation example for the course: ABS . . . 5

1 Black-box non-parametric identification of I/O systems using state-space models 8

1.1 Representations . . . 9

1.1.1 Representation #1: state-space . . . 9

1.1.2 Representation #2: transfer-function . . . 10

1.1.3 Representation #3: convolution of the input with the Impulse Response (IR) . . . 11

1.2 Transformation between representations . . . 11

1.2.1 State Space to Transfer Function . . . 12

1.2.2 Transfer Function to State Space . . . 12

1.2.3 Transfer Function to Impulse Response . . . 13

1.2.4 Impulse Response to Transfer Function . . . 14

1.2.5 State Space to Impulse Reponse . . . 14

1.2.6 Summary of transformations . . . 15

1.3 Subspace-based State Space System Identification (4SID) . . . 15

1.4 Fundamental concepts of observability and controllability . . . 17

1.4.1 Fully Observable . . . 17

1.4.2 Fully Controllable . . . 17

1.5 Hankel matrix of order n . . . 19

1.6 Algorithm to obtain $\hat{F}$, $\hat{G}$, $\hat{H}$ from a noise-free measured IR . . . 19

1.7 Real problem . . . 20

1.8 4SID procedure (with noise) . . . 21

2 Parametric black-box system identification of I/O system using a frequency domain approach 29

2.1 Comparison between time domain (ARMAX) and frequency domain parametric methods . . . 37

3 Kalman Filter (software sensing in feedback) 38

3.1 Motivation and Goals . . . 38

3.2 Kalman Filter on Basic Systems . . . 42
3.2.1 Detailed description of Basic System . . . 42
3.2.2 Basic Solution . . . 44
3.2.3 Block-scheme representation of K.F. . . . 45
3.3 Extensions of Basic Problem of Basic System . . . 46
3.3.1 Exogenous input . . . 46
3.3.2 Multi-step Prediction . . . 47
3.3.3 Filter ($\hat{x}(t|t)$) . . . 47
3.3.4 Time-varying systems . . . 48
3.3.5 Non Linear Systems . . . 48
3.4 Asymptotic Solution of K.F. . . . 48
3.4.1 Extension to Non-Linear systems . . . 58
3.4.2 Direct optimization of gain $K$ . . . 65

**4 Software-sensing with Black-Box Methods 68**
4.1 Linear Time Invariant Systems . . . 69
4.2 Comparison between K.F. and B.B. software sensing . . . 70
4.3 Non-linear Systems . . . 70

**5 Gray-box System Identification 76**
5.1 Using Kalman Filter . . . 76
5.2 Using Simulation Error Method . . . 79
5.3 Conclusion . . . 82

**6 Minimum Variance Control 84**
6.1 General solution . . . 90
6.1.1 Stability check . . . 91
6.1.2 Performance analysis . . . 92
6.2 Main limits of Minimum Variance Controllers . . . 93

**7 Recursive Identification 95**
7.1 Least square . . . 95
7.2 Recursive Least Square . . . 96
7.2.1 First form . . . 97
7.2.2 Second form . . . 98
7.2.3 Third form . . . 98
7.3 Recursive Least Square with Forgetting Factor . . . 99

**8 Appendix: Discretization of an analog system 101**

# 0 Introduction

Sergio
Savaresi
16/04/2020

## 0.1 General topics of MIDA course

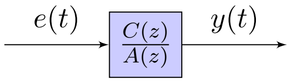
- Collect digitally data from real systems
- Build black-box (gray-box) models from data, with emphasis on
  - Dynamic systems
  - Control/automation-oriented applications
- Purpose of modelling (area of machine leasing focusing on "control")
  - Prediction
  - Software-sensing
  - Modelling for control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

### 0.1.1 Super summary of MIDA 1

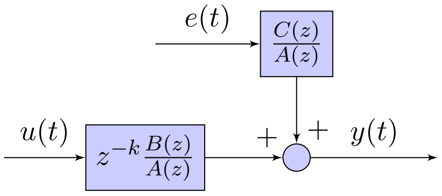
The focus is on *time series* (output-only systems) and *input/output* (I/O) systems.

Models used in MIDA1:

- ARMA models for T.S.
- ARMAX models for I/O systems



ARMA model



ARMAX model

The model is indicated as $\mathcal{M}(\theta)$ where $\theta$ is the parameter vector, the coefficients of $A(z)$, $B(z)$, $C(z)$.

A **parametric identification method** has been used: the *performance index is defined*

**Definition 1.** $J(\theta) = \frac{1}{N}\sum_{t=1}^{N}\left(y(t) - \hat{y}(t|t-1,\theta)\right)^2$

Which is the variance of the *prediction error* made by the model. The optimal $\theta$ is $\hat{\theta}_N = \arg\min_\theta J(\theta)$

### 0.1.2 MIDA 2

The focus is on I/O systems (more close to real applications than T.S.).

**Chapter 1** non-parametric (direct/constructive) black-box identification of I/O systems using state-space models

**Chapter 2** parametric identification fo black-box I/O systems, with a frequency-domain approach

**Chapter 3** Kalman-filter for Sw-sensing using feedback on white-box models

**Chapter 4** black-box methods for SW-sensing without feedback

**Chapter 5** gray-box system identification using Kalman-filter and using *simulation-error methods* (S.E.M,)

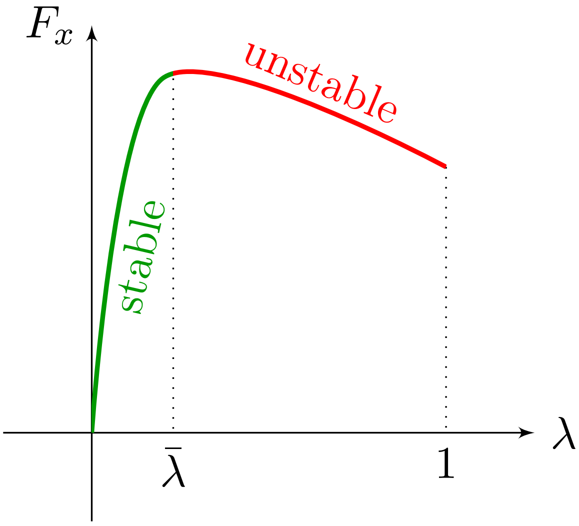
**Chapter 6** Minimum-Variance Control (M.V.C.), design of optimal feedback controllers using the theory background of the MIDA course

**Appendix** Recursive (online) implementation of algorithms for system identification

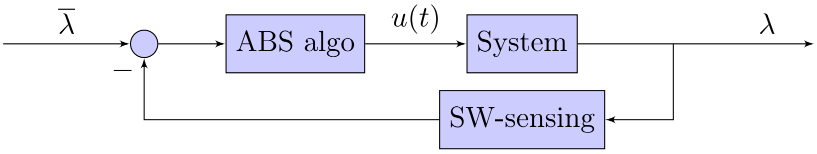
## 0.2 Motivation example for the course: ABS

**Definition 2.** ***Slip*** *of the wheel:* $\lambda = \frac{v-\omega r}{v}$

During a break $0 \leq \lambda \leq 1$ (from free rolling wheel to locked wheel).



Relation between $\lambda$ and the breaking force.



ABS system

The problem can be divided into subproblems:

- Model of the system
- SW-estimation of $\lambda$ ($v$ is not directly measurable, so $\lambda$ cannot be computed)
- Design of the ABS control algorithm

Why black-box modelling? The control variable $v$ (the voltage to the actuator) controls a complex systems from the actuator to $\lambda$. The system can be seen as a chain of components:

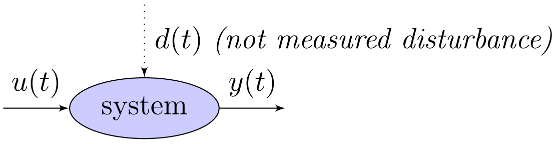
- Current dynamics and electric motor

- Position dynamics of the actuator
- Dynamics of the hydraulic circuit of the break system
- Tire dynamics
- Wheel rotational dynamics
- Vehicle full dynamics

White box (physical) modelling: write the equations from *first principles*.

Black box modelling: experiment → collect data → build model. Using only I/O measured data we can *learn* a mathematical model of the I/O behavior of the system.

# 1 Black-box non-parametric identification of I/O systems using state-space models



**Measured data**

$$\begin{aligned} &\{u(1), u(2), \dots, u(N)\} \quad \text{(input)} \\ &\{y(1), y(2), \dots, y(N)\} \quad \text{(output)} \end{aligned}$$

**Remark: general path of a parametric identification methods**

1. Collect data: $\{u(1), u(2), \dots, u(N)\}$, $\{y(1), y(2), \dots, y(N)\}$
2. Select **a-priori** a class/family of parametric models: $\mathcal{M}(\theta)$
3. Select **a-priori** a performance index (it gives an order to the quality of the models)
4. Optimization step (minimize $J(\theta)$ w.r.t $\theta$): $\hat{\theta}_N = \arg\min_\theta J(\theta) \rightarrow$ optimal model $\mathcal{M}(\hat{\theta}_N)$

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

$\mathcal{M}(\theta_1)$ is better than $\mathcal{M}(\theta_2)$ if $J(\theta_1) < J(\theta_2)$.

In this chapter we are presenting a totally different system identification approach: **not parametric**.

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

## 1.1 Representations

### 1.1.1 Representation #1: state-space

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

Where $F$, $G$, $H$ and $D$ are matricies defined as follows:

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

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

Assuming 1 input and 1 output, it can be extended for multiple inputs and outputs. Usually $D = 0$ for *strictly-proper systems.*

**Remark: S.S representation is not unique**

$F_1 = TFT^{-1}$, $G_1 = TG$, $H_1 = HT^{-1}$, $D_1 = D$ for any invertible matrix $T$. The system $\{F, G, H, D\}$ is equivalent to $\{F_1, G_1, H_1, D_1\}$.

**Example**

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

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

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

### 1.1.2 Representation #2: transfer-function

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

$W(z)$ is a rational function of the $z$ operator: it's a *digital filter*.

It's very easy to move from T.F. representation to a time domain description of the system.

**Example**

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

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

$$y(t) = \underbrace{-\frac{1}{6}y(t-1) - \frac{1}{8}y(t-2)}_{\text{old values of } y(t)} + \underbrace{\frac{1}{2}u(t-1) + \frac{1}{4}u(t-2)}_{\text{old values of input}}$$

**Remark: Notational remark**

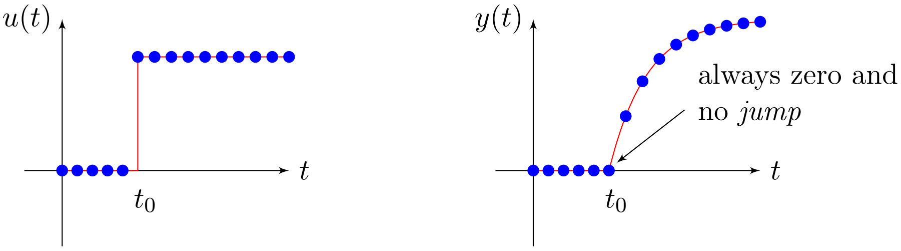
$W(z) = \dfrac{z^{-1}}{1+\frac{1}{3}z^{-1}}$ is called an IIR (*Infinite Impulse Response*) filter.

$W(z) = z^{-1} + \dfrac{1}{2}z^{-2} + \dfrac{1}{4}z^{-3}$ is called a FIR (*Finite Impulse Response*) filter.

Sergio
Savaresi
20/04/2020

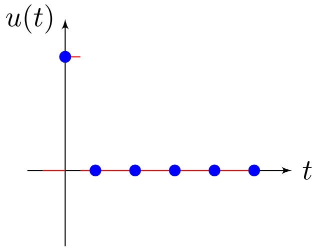
**Remark: Strictly proper systems**

Notice that for strictly proper systems the delay $k \geq 1$

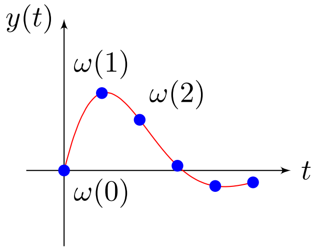


### 1.1.3 Representation #3: convolution of the input with the Impulse Response (IR)

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



Input



Output

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

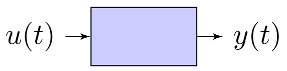
It can be proven that the input-output relationship from $u(t)$ to $y(t)$ can be written as

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

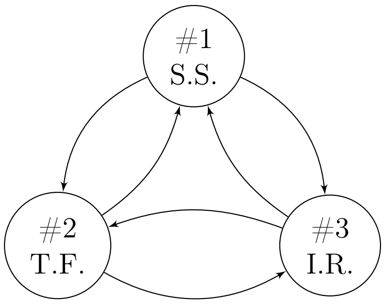
It can be rewritten as follows

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

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



## 1.2 Transformation between representations



Transformations between representations

### 1.2.1 State Space to Transfer Function

Consider a strictly proper system with the following state space representation:

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

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

$$W(z) = H(zI - F)^{-1}G$$

**Example**

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

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

Notice that 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.

### 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 model.

Note that the state space representation is not unique, so from a single transfer function we can get infinite different equivalent state space models.

### Control realization

We assume that the system is strictly proper and that the denominator is monic.

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

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

$$F = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ 0 & 0 & \cdots & 0 & 1 \\ -a_n & -a_{n-1} & & \cdots & -a_1 \end{bmatrix} \quad G = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix} \quad H = \begin{bmatrix} b_{n-1} & b_{n-2} & \cdots & b_0 \end{bmatrix} \quad D = 0$$

**Example**

Consider the transfer function $W(z)$

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

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 $\infty$-long division between the numerator and denominator of $W(z)$

**Example**

Consider the transfer function

$$W(z) = \frac{1}{z - \frac{1}{2}} = \frac{z^{-1}}{1 - \frac{1}{2}z^{-1}} = 0z^{-0} + 1z^{-1} + \frac{1}{2}z^{-2} + \frac{1}{4}z^{-3} + \cdots$$

Thus the IR is $\omega(0) = 0$, $\omega(1) = 1$, $\omega(2) = \frac{1}{2}$, $\omega(3) = \frac{1}{4}$, ...

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

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

Given this, it can be proven that

$$W(z) = \mathcal{Z}\left(\omega(t)\right) = \sum_{t=0}^{\infty} \omega(t)z^{-t}$$

This means that the transfer function of a system is the $\mathcal{Z}$-transform of a special signal, 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 the impulse response must be noise-free. Thus, this transformation is only theoretical.

### 1.2.5 State Space to Impulse Reponse

Consider the following state space model, with initial conditions $x(0) = 0$ and $y(0) = 0$

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

We have that

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

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

$$\begin{aligned} x(3) &= Fx(2) + Gu(2) = F^2Gu(0) + FGu(1) + Gu(2) \\ y(3) &= Hx(3) = HF^2Gu(0) + HFGu(1) + HGu(2) \end{aligned}$$

This can be generalized to

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

Thus, the impulse response is

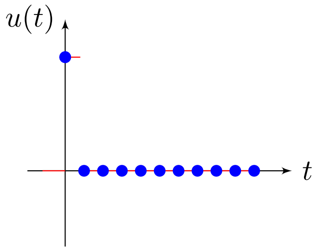
$$\omega(t) = \begin{cases} 0 \text{ if } t = 0 \\ HF^{t-1}G \text{ if } t > 0 \end{cases}$$

### 1.2.6 Summary of transformations

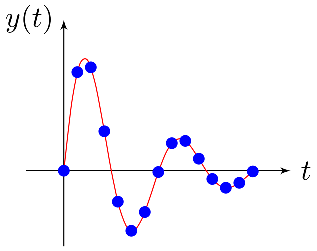
Notice that the IR representation is very easy to obtain experimentally, since we only need to measure the system response to the impulse signal. However, given the IR representation, it is difficult to get to the other representations, since the transformation between from IR to TF is only theoretical.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

## 1.3 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.*



Input



Output

The fundamental idea is that it is very simple to make this experiment, that is to measure the output of system given an impulse signal as the input. The real problem is to identify a model $\left\{\hat{F}, \hat{G}, \hat{H}\right\}$ starting from $\{\omega(0), \omega(1), \omega(2), \cdots\}$.

We will see the solution of this problem in two steps:

1. IR measurement is assumed to be noise-free. This is easier with respect to the original problem, but it is also not realistic.
2. IR is measured with noise

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

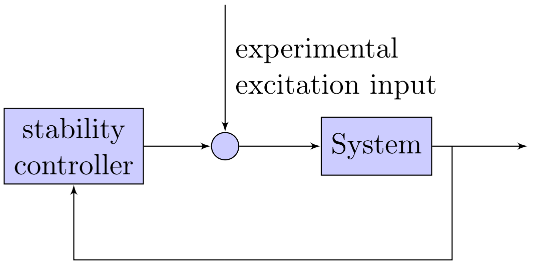
- $\widetilde{\omega}(t)$ is the measured noisy IR
- $\omega(t)$ is the "true" noise-free IR
- $\eta(t)$ is the measurement noise (e.g. WN)

**Remark**

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

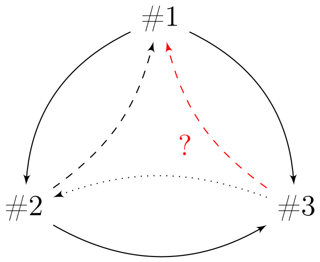
**Remark: Unstable system**

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.



Closed loop system

Sergio
Savaresi
21/04/2020



## 1.4 Fundamental concepts of observability and controllability

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

### 1.4.1 Fully Observable

The system is fully observable (from the output) if and only if the observability matrix is full rank:

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

### 1.4.2 Fully Controllable

The system is fully controllable (from the input) if and only if the controllability matrix is full rank:

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

$R$ is also called *reachability* matrix.

**Remark**

**Observability** we can observe the state from the output sensors.

**Controllability** we can control/move/influence the state using the input signal.

**Example**

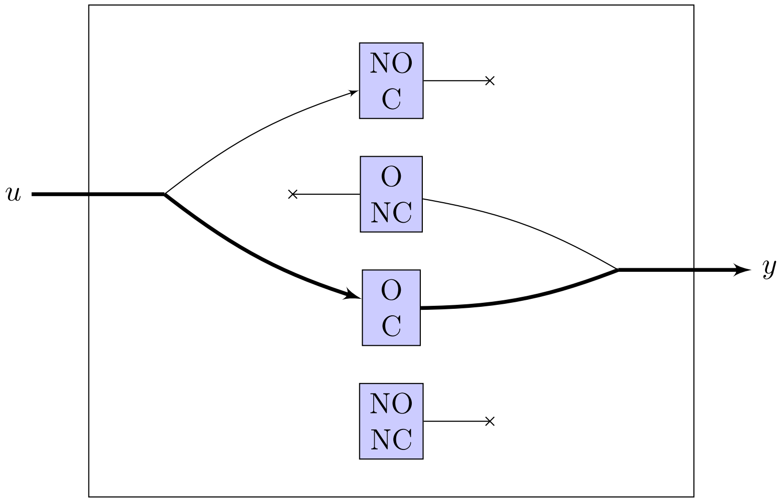
$$\begin{cases} x_1(t+1) = \frac{1}{2}x_1(t) + u(t) \\ x_2(t+1) = \frac{1}{3}x_2(t) \\ y(t) = \frac{1}{4}x_1(t) \end{cases} \qquad F = \begin{bmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{3} \end{bmatrix} \qquad H = \begin{bmatrix} \frac{1}{4} & 0 \end{bmatrix} \qquad G = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

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

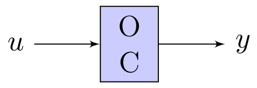
$$R = \begin{bmatrix} G & FG \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 1 & \frac{1}{3} \end{bmatrix} \qquad \operatorname{rank} R = 1 < n = 2 \qquad \implies \qquad \text{not fully controllable}$$

**Remark: 4 sub-systems**

Each system can be internally seen as 4 sub-systems as follows:



Which externally is equivalent to a systems like this:



## 1.5 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** that we need IR up to time $2n-1$.

**Note** it starts from $\omega(1)$ and not $\omega(0)$.

**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

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

Where $O$ is the observability matrix and $R$ is the reachability matrix.

## 1.6 Algorithm to obtain $\hat{F}$, $\hat{G}$, $\hat{H}$ from a noise-free measured IR

**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 $\operatorname{rank} H_n = n$ and $\operatorname{rank} H_{n+1} = n$. With this procedure we know the order of the system.

**Step 2** Take $H_{n+1}$ (with $\operatorname{rank} H_{n+1} = n$), 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:

$$\begin{aligned} \hat{G} &= R_{n+1}(\texttt{:;1}) \quad \text{first column} \\ \hat{H} &= O_{n+1}(\texttt{1;:}) \quad \text{first row} \end{aligned}$$

Define $O_1$ as $O_{n+1}$ without the last row, and $O_2$ as $O_{n+1}$ without the first row. $O_1$ and $O_2$ are $n \times n$ matrices. This is called *shift-invariance* property.

**Note** that $O_1$ is invertible since it's an observability matrix and the subsystem is fully observable. Moreover $O_1 F = O_2$, so $\hat{F} = O_1^{-1} O_2$.

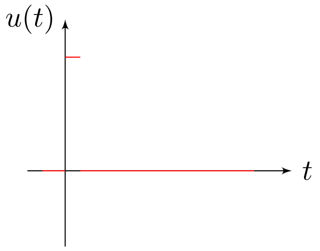
In conclusion in a simple and constructive way we have estimated a State Space model of the system $\{\hat{H}, \hat{G}, \hat{F}\}$ starting from measured IR, using only $2n+1$ samples of IR.

**Remark**

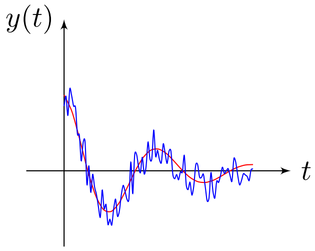
If the measurement is noisy all this process is useless.

## 1.7 Real problem

The measurements of IR are noisy: $\tilde{\omega}(t) = \omega(t) + \eta(t)$.



Input



Output

## 1.8 4SID procedure (with noise)

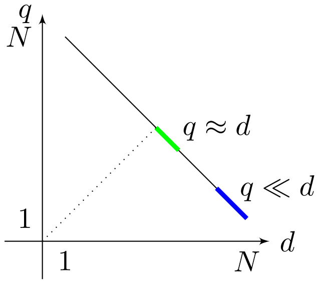
**Step 1** Build the Hankel matrix from data using "one-shot" all the available $N$ data points.

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

$\tilde{H}_{qd}$ is a $q \times d$ matrix. **Note** that $q+d-1$ must equal to $N$ so that we use all the data set.

**Remark: Choice of $q$ and $d$**

Hypothesis: $q < d$ so that $q+d-1 = N$, so $q = N+1-d$.



If $q \approx d$ the method has better accuracy. If $q \ll d$ it's computationally less intensive.

If $0.6d < q < d$ we get to a good enough result.

Sergio
Savaresi
21/04/2020

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

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

$\tilde{U}$ and $\tilde{V}$ are unitary matrices. A matrix $M$ is unitary if:

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

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

Where $\sigma_1$, $\sigma_2$, ..., $\sigma_q$ are the singular values of $\tilde{H}_{qd}$. Those are real, positive numbers, sorted in decreasing order ($\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_q$).

**Remark**

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.

**Remark**

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

**Remark: How to compute SVD**

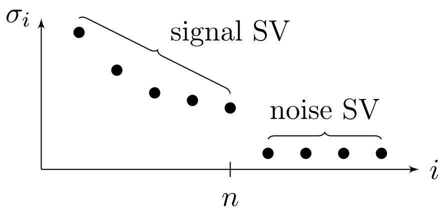
The optimal numerical computation is not trivial. Use `svd(M)` in Matlab.

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

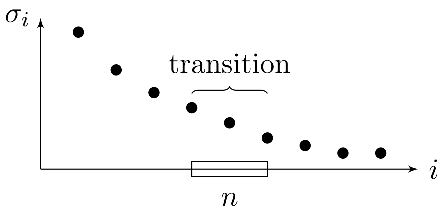
$$\underbrace{\tilde{H}_{qd}\tilde{H}_{qd}^T}_{q\times q} = \tilde{U}\tilde{S}\tilde{S}^T\tilde{U}^T$$

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

**Step 3** Plot the singular values and cut-off the 3 matrices.



Ideal case

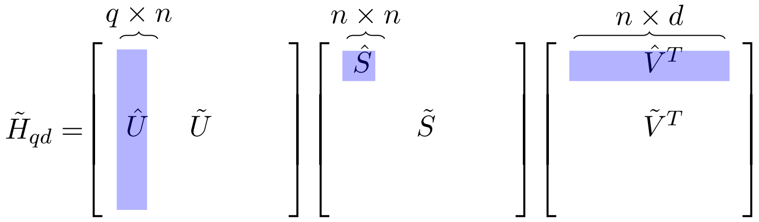


Real case

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, 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 $\tilde{U}$, $\tilde{S}$ and $\tilde{V}^T$:



$$\tilde{H}_{qd} = \underbrace{\hat{U}\hat{S}\hat{V}^T}_{\hat{H}_{qd}} + H_{res,qd} \qquad \text{rank}\, \tilde{H}_{qd} = q \quad \text{rank}\, \hat{H}_{qd} = n \quad \text{rank}\, H_{res,qd} = q$$

From $\tilde{H}_{qd}$ to $\hat{H}_{qd}$ the rank is hugely reduced.

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

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

Where $\hat{S}^{\frac{1}{2}}$ is the matrix with elements the square roots of the elements of $\hat{S}$.

$$\hat{O} = \hat{U}\hat{S}^{\frac{1}{2}} \qquad \hat{R} = \hat{S}^{\frac{1}{2}}\hat{V}^T \qquad \Longrightarrow \qquad \hat{H}_{qd} = \hat{O}\hat{R}$$

We can view $\hat{O}$ as the extended observability matrix and the $\hat{R}$ the extended reachability matrix of the system. We can estimate $\hat{H}$ with the first row of $\hat{O}$ and $\hat{G}$ with the first column of $\hat{R}$.

What about the estimation of $\hat{F}$? Consider for example $\hat{O}$ and use the *shift-invariance* property. Define $\hat{O}_1$ as $\hat{O}$ without the last row, and $\hat{O}_2$ as $\hat{O}$ without the first row.

Therefore $\hat{O}_1 \cdot \hat{F} = \hat{O}_2$, but $\hat{O}_1$ is not a square matrix so it's not invertible. In this case we can use the approximate *least-square* solution of this linear system.

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).

In the third case we can use an approximate solution using the least-squares method, which for a generic system is as follows:

$$\begin{aligned} Ax &= B \\ A^TAx &= A^TB \implies \hat{X} = \underbrace{(A^TA)^{-1}A^T}_{A^+}B \end{aligned}$$

$A^+$ is called *pseudo-inverse*, which is "surrogate inverse" when $A$ is rectangular.

Using the pseudo-inverse matrix:

$$\hat{O}_1\hat{F} = \hat{O}_2 \qquad \hat{O}_1^T\hat{O}_1\hat{F} = \hat{O}_1^T\hat{O}_2 \qquad \hat{F} = \left(\hat{O}_1^T\hat{O}_1\right)^{-1}\hat{O}_1^T\hat{O}_2$$

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

**Remark**

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 $\tilde{H}_{qd}$.

**Example: Rank reduction**

In general there are infinite ways to make a rank reduction.

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

It's not the optimal rank reduction matrix, but it factors out a matrix with lower rank.

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: $\tilde{H}_{res,qd}$ is the minimum possible (in the sense of the *Frobenius norm*).

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

**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.

Sergio Savaresi 23/04/2020

**Exercise: similar to an exam exercise**

Consider the following S.S. model

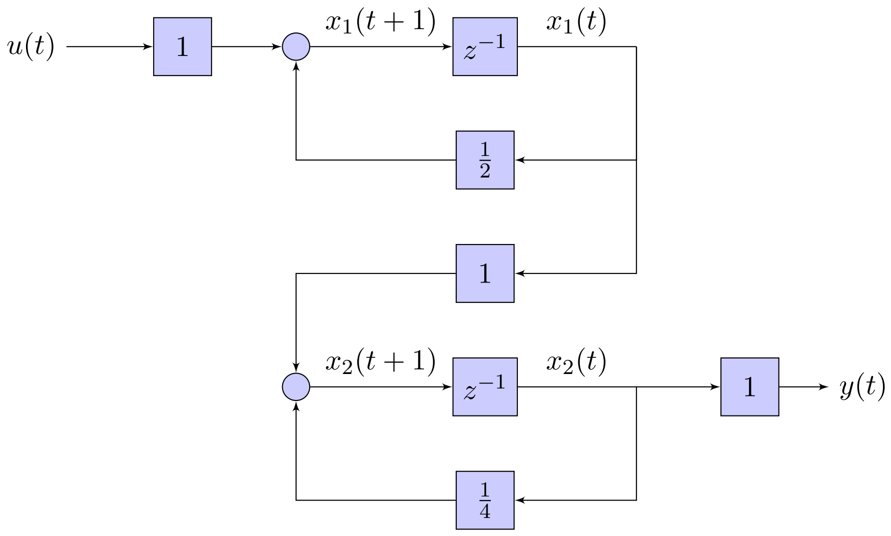
$$F = \begin{bmatrix} \frac{1}{2} & 0 \\ 1 & \frac{1}{4} \end{bmatrix} \qquad G = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \qquad H = \begin{bmatrix} 0 & 1 \end{bmatrix} \qquad D = 0$$

The system has grade $n = 2$ and is single-input single-output.

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

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

**Question** Write the block scheme of the S.S. representation of the system.



By visual inspection the system is fully observable and fully controllable.

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

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

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

The extended $(n+1)$ $O_3$ and $R_3$:

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

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

**Question** Compute the transfer function representation.

First method: direct manipulation of S.S. equations

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

$$\begin{aligned} zx_1(t+1) - \frac{1}{2}x_1(t) = u(t) \qquad &\Longrightarrow \qquad x_1(t) = \frac{1}{z-\frac{1}{2}}u(t) \\ zx_2(t) - \frac{1}{4}x_2(t) = \frac{1}{z-\frac{1}{2}}u(t) \qquad &\Longrightarrow \qquad x_2(t) = \frac{1}{(z-\frac{1}{4})(z-\frac{1}{2})}u(t) \\ y(t) = \frac{1}{(z-\frac{1}{4})(z-\frac{1}{2})}u(t) \qquad & \end{aligned}$$

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

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** 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** Compute the first 6 values (including $\omega(0)$) of I.R.

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

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

**Question** Build the Hankel matrix and stop when the rank is not full.

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

$H_3$ is not full rank so the order of the system is 2. Notice that $O_3 R_3 = H_3$.

# 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:

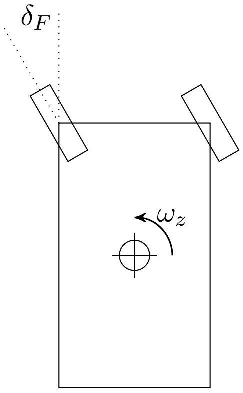
1. Experiment design and data pre-processing. A special type of experiment and data pre-processing is needed.
2. Selection of parametric model class ($\mathcal{M}(\theta)$)
3. Definition of a performance index ($J(\theta)$). A new special performance index is needed.
4. Optimization ($\hat{\theta} =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

Sergio Savaresi 27/04/2020

The general ideal of the method is:

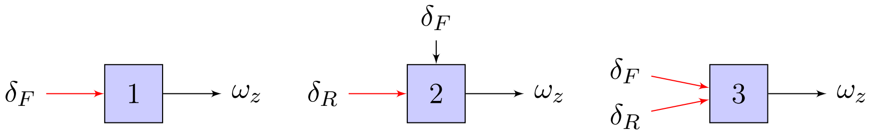
- Make a set of "single sinusoid" ("single-tune") excitation experiments
- From each experiment estimate a single point of the frequency response of the system
- Fit the estimated and modeled frequency response to obtain the optimal model

**Example: Car steer dynamics**



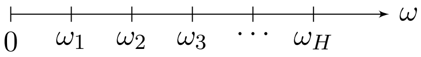
This is the dynamic relationship linking the input (steer) to the output (yaw-rate). This kind of relationship is very important for stability control design and autonomous car.

There are 3 possible situations:



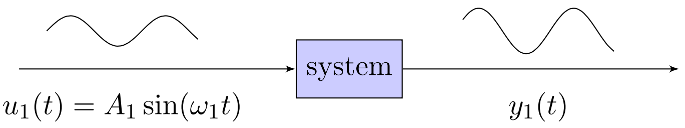
1. The control variable is the front steer: application is autonomous car
2. The driver controls $\delta_F$ which is a measurable disturbance, the system controls the rear steer
3. Both $\delta_R$ and $\delta_F$ are control variables: application high performance autonomous car

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

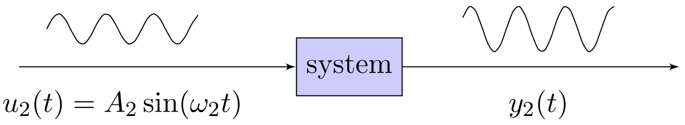


The maximum frequency is $\omega_H$. We have $\{\omega_1, \omega_2, \cdots, \omega_H\}$ usually evenly spaced ($\Delta\omega$ is constant). $\omega_H$ must be selected according to the bandwidth of the control system.

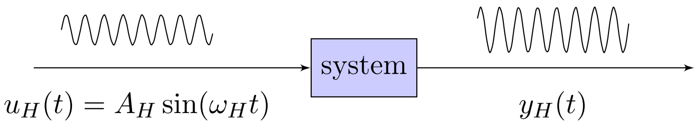
We make $H$ independent experiments.



Experiment #1



Experiment #2



Experiment #$H$

**Remark**

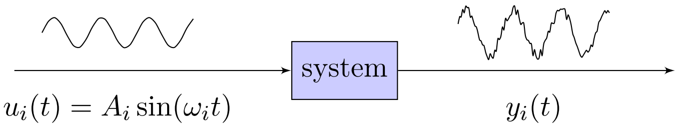
The amplitudes $A_1$, $A_2$, ..., $A_H$ can be equal (constant) or, more frequently, they decrease as the frequency increases to fulfill the power constraint on the input.

**Example** $\delta(t)$ is the steering angle (moved by an actuator). The requested steer torque is proportional to $\delta$: $T(t) = K\delta(t)$. 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.

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

Focusing on the $i$-th experiment.



**Remark**

If the system is LTI (linear time-invariant), the frequency response theorem says the if the input is a sine input of frequency $\omega_i$ the output must be a sine with frequency $\omega_i$.

However $y_i(t)$ in real applications is not a perfect sinusoid.

- Noise on output measurements
- Noise on the system (not directly on the output)
- (Small) non-linear effects (that we neglect)

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

The model of the output signal is

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

There are 2 unknowns: $B_i$ and $\phi_i$ (or $a_i$ and $b_i$). We try to estimate $a_i$ and $b_i$ since they are *linear*.

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

$$\{\hat{a}_i, \hat{b}_i\} = \arg\min_{\{a_i, b_i\}} J_N(a_i, b_i)$$

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

$J_N$ is a quadratic function of $a_i$ and $b_i$, so we can solve the problem explicitly.

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

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

At this point we prefer to go back to a *sin-only* form ($B_i$, $\phi_i$):

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

$$\begin{aligned} \hat{B}_i \cos(\hat{\phi}_i) &= \hat{a}_i \\ \hat{B}_i \sin(\hat{\phi}_i) &= \hat{b}_i \end{aligned}$$

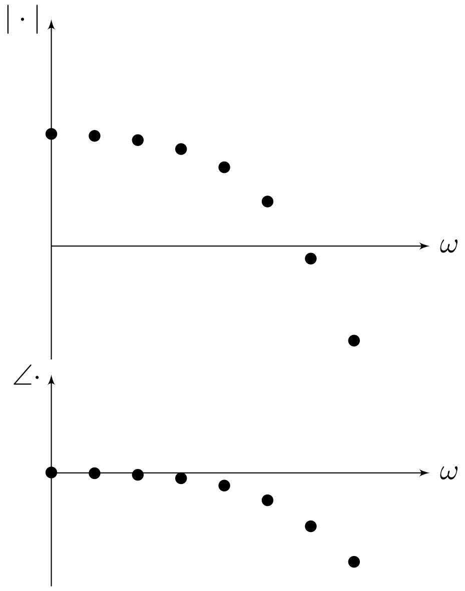
$$\frac{\hat{b}_i}{\hat{a}_i} = \frac{\sin\hat{\phi}_i}{\cos\hat{\phi}_i} = \tan(\hat{\phi}_i) \qquad \hat{\phi}_i = \arctan\left(\frac{\hat{b}_i}{\hat{a}_i}\right)$$

$$\hat{B}_i = \frac{\frac{\hat{a}_i}{\cos\hat{\phi}_i} + \frac{\hat{b}_i}{\sin\hat{\phi}_i}}{2}$$

Repeating the experiment and pre-processing for the $H$ experiments we obtain

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

We obtain $H$ complex numbers, the estimated $H$ points of the frequency response of the transfer function $W(z)$ from the input $u(t)$ to the output $y(t)$ of the system.



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** Model class selection (T.F.)

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

**Remark: 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).

**Step 3** New performance index (frequency domain).

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

**Step 4** Optimization

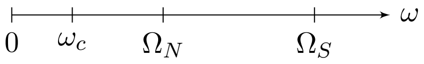
$$\hat{\theta} = \arg\min_{\theta} J_H(\theta)$$

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

**Remark: Frequency bandwidth selection $\omega_H = ?$**

Theoretically the standard best solution should be $H$ points distributed uniformly from 0 to $\Omega_N$ (Nyquist).

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



$\omega_c$ is the expected cut-off frequency of the closed system. A rule of thumb: $\omega_H \approx 3\omega_c$

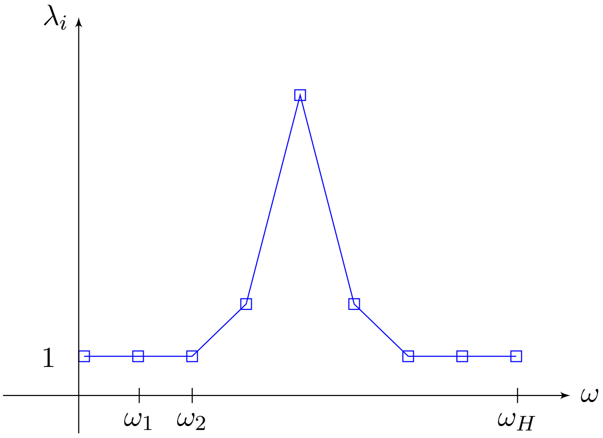
**Example** The ESC (electronic stability control) has an expected bandwidth of $\omega_c \approx 2$Hz, so $\omega_H \approx 6$Hz.

Sergio
Savaresi
30/04/2020

**Remark: Emphasis on special frequency range**

In some cases, between $\omega_1$ and $\omega_H$, we want to be more accurate in system identification in some frequency regions (typically around cut-off-frequency, around resonances).

We can use different weights for different frequencies.



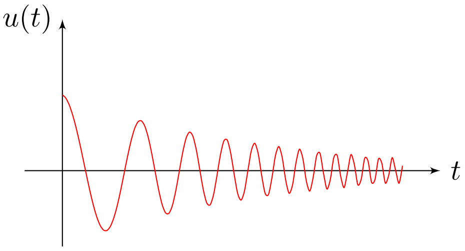
The performance index can be redefined:

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

Another *trick*: more dense $\omega_i$ 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.



Slowing-varying sinusoid with increasing frequency and decreasing amplitude.

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

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

We can fit the estimated $\hat{W}(e^{j\omega})$ with the model frequency response $W(e^{j\omega}, \theta)$ in the performance index. This experiment is quicker but has usually a lower signal-to-noise-ration.

## 2.1 Comparison between time domain (ARMAX) and frequency domain parametric methods

### Frequency domains

**Pro** Robust and very reliable. We put a lot of energy on each sinusoid. The frequency response estimation is very reliable.

**Pro** Intuitive (easy to understand)

**Pro** Consistent with control-design methods (usually in frequency domain)

**Cons** More demanding for the experiment

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

# 3 Kalman Filter (software sensing in feedback)

In MIDA1 we have mostly used I/O transfer function representations:

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

Kalman filter theory is fully based on S.S. representation.

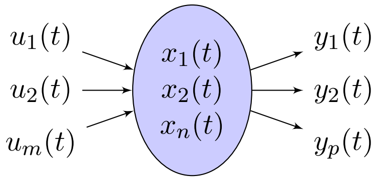
$$\begin{cases} x(t+1) = Fx(t) + Gu(t) + v_1(t) & v_1 \sim WN \\ y(t) = Hx(t) + \cancel{Du(t)} + v_2(t) & v_2 \sim WN \end{cases}$$

## 3.1 Motivation and Goals

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

- $k$-steps ahead prediction of output: $\hat{y}(t+k|t)$ (already solved in MIDA1 for ARMAX).
- $k$-steps ahead prediction of state: $\hat{x}(t+k|t)$ (at time $t$ we have available $y(t)$, $y(t-1)$, …, $u(t)$, $u(t-1)$, …).
- Find the filter of the state: $\hat{x}(t|t)$ (given data of $y(t)$ and past, and $u(t)$ and part, the estimation is made at the same time). In practice it's *software-sensing*, most important problem solved by Kalman filter (ideed K.F. is named *filter*).
- Gray box system identification (see chapter 5). We have a recorded data set and the model structure with some unknown parameters.

Dynamical systems have this layout:



MIMO system with $m$ inputs, $p$ outputs and $n$ states.

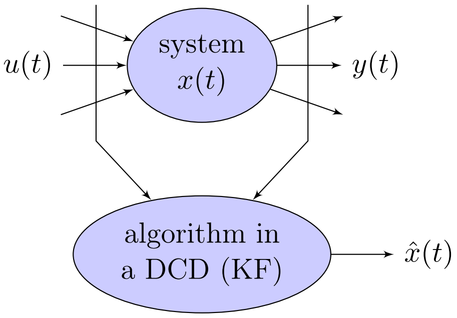
**Key problem** usually $p \ll n$, physical sensors are much less than system states.

- Cost
- Cables, power supply
- Maintenance (faults, degradation)

It is useful to have full "measurement" of states because:

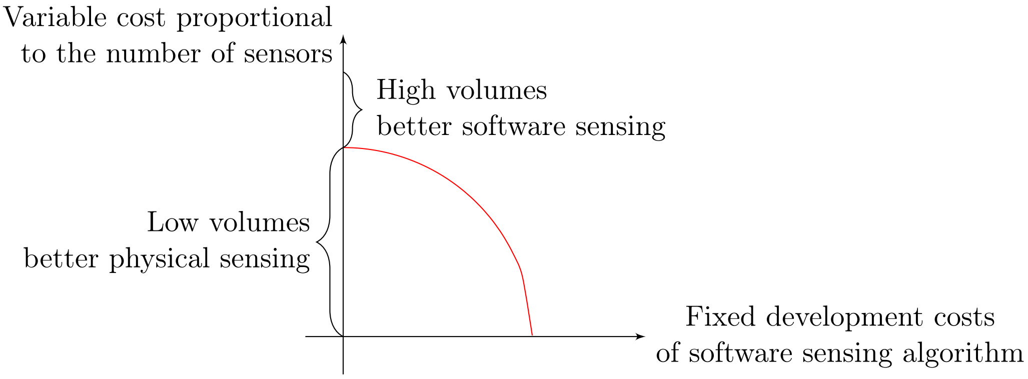
- Control design (state feedback design)
- Monitoring (fault detection, predictive maintenance, ...)

Software sensing (also called virtual sensing):



**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



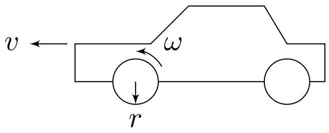
Break-even analysis

Sergio
Savaresi
04/05/2020

### Key questions for software sensing

- Is software-sensing feasible? Test is the observability of the states from measured outputs.
- Quality of estimation error (*noise of measurement* for the software sensor).

**Example: Slip estimation for ABS/traction control**



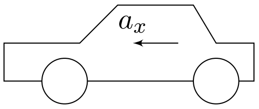
**Problem** Estimation of $v$.

Measure of $v$ can be done with:

- Optical sensor
- GPS

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.



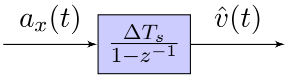
$$\hat{v} = \int a_x(t)dt \qquad \Longleftrightarrow \qquad a_x(t) \to 1/s \to \hat{v}(t)$$

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

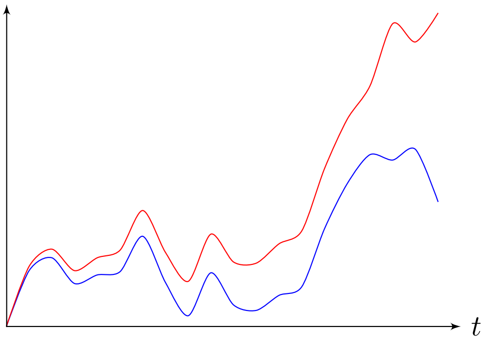
$$\frac{d}{dt}v(t) = a_x(t) \qquad \frac{dv(t)}{dt} \approx \frac{v(t+1) - v(t)}{\Delta T_s} = a_x(t)$$

Where $\Delta T_s$ is the sampling interval (e.g. 10ms).

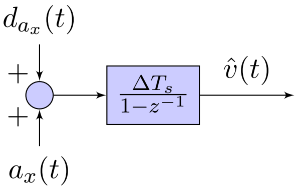
$$\hat{v}(t) = \hat{v}(t-1) + \Delta T_s a_x(t-1)$$



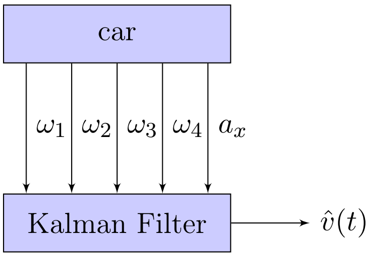
Unfortunately the measured signal is not $a_x(t)$ but $a_x(t) + d_{a_x}(t)$.



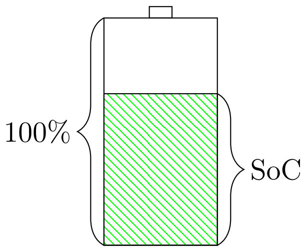
Integrating noise generates a *drift*. Integrator is not an asymptotic stable system.

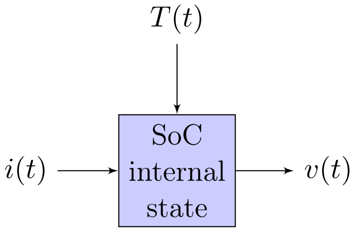


**Solution** Use a Kalman Filter.



**Example: State of charge estimation of a battery**





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

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

After the description of the solution of Kalman Filter for the basic system, we make the extensions to a more general system, in particular with $Gu(t)$.

### 3.2.1 Detailed description of Basic System

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

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

System with $n$ states, ($m$ inputs) and $p$ outputs.

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

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

$v_1(t)$ is called state noise or model noise. It accounts for the modelling errors of the state equation of the system.

- $E[v_1(t)] = \mathbf{0}$
- $E[v_1(t) \cdot v_1(t)^T] = V_1$, where $V_1$ is an $n \times n$ covariance matrix (symmetric and semi-definite positive)
- $E[v_1(t) \cdot v_1(t-\tau)^T] = 0 \quad \forall t \forall \tau \neq 0$

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

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

$v_2(t)$ is called output noise or measurement (or sensor) noise.

- $E[v_2(t)] = \mathbf{0}$
- $E[v_2(t) \cdot v_2(t)^T] = V_2$, where $V_2$ is a $p \times p$ covariance matrix (symmetric and **definite** positive)
- $E[v_2(t) \cdot v_2(t-\tau)^T] = 0 \quad \forall t \forall \tau \neq 0$

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

$$E[v_1(t) \cdot v_2(t-\tau)^T] = \underbrace{V_{12}}_{n \times p} = \begin{cases} 0 & \text{if } \tau \neq 0 \\ \text{can be non-zero} & \text{if } \tau = 0 \end{cases}$$

They can be correlated but only at the same time (in practice $V_{12} = 0$ is the most common assumption).

Since the system $S$ is dynamic we need to define the initial conditions:

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

If $P_0 = 0$ 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:

$$x(1) \perp v_1(t) \qquad x(1) \perp v_2(t)$$

### 3.2.2 Basic Solution

$$\begin{aligned}
&\hat{x}(t+1|t) = F\hat{x}(t|t-1) + K(t)e(t) && \text{state equation} \\
&\hat{y}(t|t-1) = H\hat{x}(t|t-1) && \text{output equation} \\
&e(t) = y(t) - \hat{y}(t|t-1) && \text{output prediction error} \\
&K(t) = \left(FP(t)H^T + V_{12}\right)\left(HP(t)H^T + V_2\right)^{-1} && \text{gain of the K.F.} \\
&P(t+1) = \left(FP(t)F^T + V_1\right) + \\
&-\left(FP(t)H^T + V_{12}\right)\left(HP(t)H^T + V_2\right)^{-1}\left(FP(t)H^T + V_{12}\right)^T && \text{difference Riccati equation}
\end{aligned}$$

These equations must be completed with 2 initial conditions (2 are dynamic equations)

$$\begin{aligned}
\hat{x}(1|0) = E[x(1)] = X_0 &\qquad \text{state equation} \\
P(1) = \text{var}[x(1)] = P_0 &\qquad \text{DRE}
\end{aligned}$$

**Remark: Structure or $K(t)$ and D.R.E**

Notice that $K(t)$ and DRE have a *blockset* structure having this form: $AP(t)B^T + N$

There are 3 different types of blocks:

$$\begin{aligned}
\text{state} &\qquad FP(t)F^T + V_1 \\
\text{output} &\qquad HP(t)H^T + V_2 \\
\text{mix} &\qquad FP(t)H^T + V_{12}
\end{aligned}$$

$$\begin{aligned}
\text{gain} &\qquad (\text{mix})(\text{output})^{-1} \\
\text{DRE} &\qquad (\text{state}) - (\text{mix})(\text{output})^{-1}(\text{mix})^T
\end{aligned}$$

**Remark: Riccati equation**

Riccati equation is a special type of nonlinear matrix difference equation.

Notice that DRE is an autonomous, non-linear, discrete time, multi-variable system, described by a non-linear difference matrix equation.

$$\text{DRE: } P(t+1) = f(P(t)) \qquad P(1) = P_0$$

**Remark: Existance of DRE**

In order to guarantee the existance of DRE for all $t$ the only critical part is the inversion of the *output* block:

$$(\underbrace{HP(t)H^T}_{\geq 0} + \underbrace{V_2}_{>0})^{-1} \qquad \text{Thanks to } V_2 > 0 \text{ it's invertible}$$

**Remark: Meaning of $P(t)$**

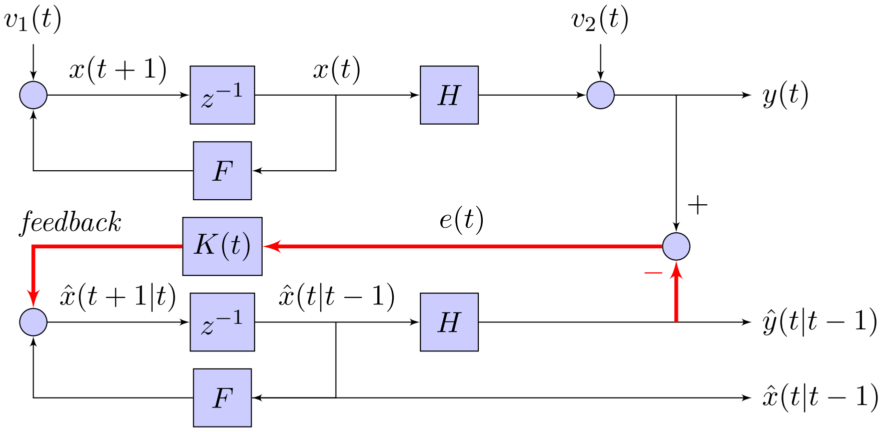
It can be proven that $P(t)$ has a very important meaning.

$$P(t) = \text{var}[x(t) - \hat{x}(t|t-1)] = E[(x(t) - \hat{x}(t|t-1))(x(t) - \hat{x}(t|t-1))^T]$$

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

Sergio Savaresi 05/05/2020

### 3.2.3 Block-scheme representation of K.F.



The idea behind Kalman Filter is simple and intuitive:

- We make a simulated replica of the system (without noises $v_1$ and $v_2$, not measurable)

- We compare the true measured output with the estimated/simulated output $\hat{y}(t|t-1)$
- We make corrections on K.F. main equation, proportional (with gain $K(t)$) to the output error in order to keep K.F. as close as possible to the system
- We extract the state estimation $\hat{x}(t|t-1)$

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

This general structure was known before Kalman Filter development, it's 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:

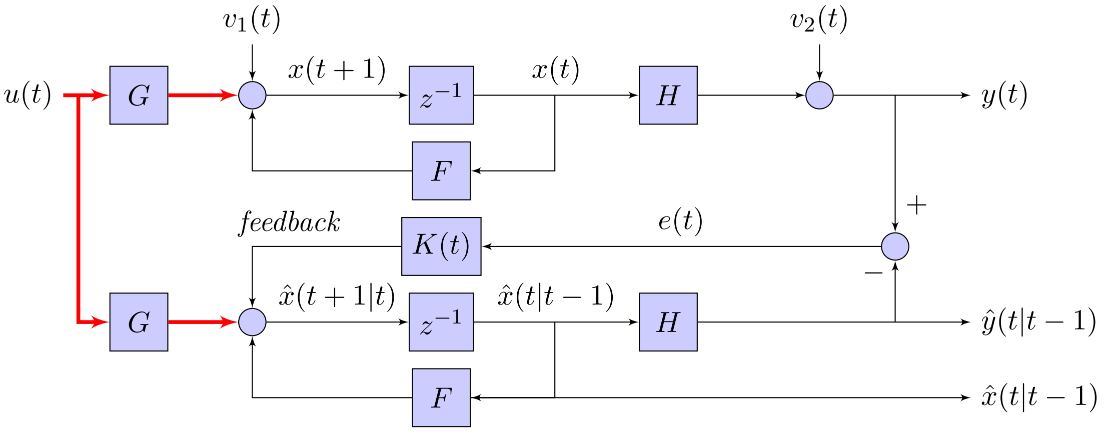
- If $K(t)$ is *too small*: the estimation is not optimal because we are *under exploiting* the information in $y(t)$
- If $K(t)$ is *too big*: risk of over-exploiting $y(t)$ and we can get noise amplification, even risk of instability

Design of 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; $V_1$ much more difficult to be designed (it's the most critical design parameter of K.F.)

## 3.3 Extensions of Basic Problem of Basic System

### 3.3.1 Exogenous input



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

### 3.3.2 Multi-step Prediction

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

$$\begin{aligned}\hat{x}(t+2|t) &= F\hat{x}(t+1|t)\\ \hat{x}(t+3|t) &= F\hat{x}(t+2|t) = F^2\hat{x}(t+1|t)\\ &\vdots\\ \hat{x}(t+k|t) &= F^{k-1}\hat{x}(t+1|t)\\ \hat{y}(t+k|t) &= H\hat{x}(t+k|t)\end{aligned}$$

### 3.3.3 Filter ($\hat{x}(t|t)$)

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

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

Kalman Filter in the filter form is:

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

**Remark**

These equations are valid under the restrictive assumption $V_{12} = 0$.

**Remark**

Gain of K.F. in prediction form:

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

Gain of K.F. in filter form:

$$K_0(t) = \left(\_P(t)H^T\right)\left(HP(t)H^T + V_2\right)^{-1}$$

### 3.3.4 Time-varying systems

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

Kalman Filter equations are exactly the same.

### 3.3.5 Non Linear Systems

This extensions is much more complicated: Extended Kalman Filter (EKF).

## 3.4 Asymptotic Solution of K.F.

Observe that K.F. is not itself an LTI system but is a LTV system, because the gain $K(t)$ is time-varying.

The fact that K.F. is a LTV system is the source of 2 problems:

- Checking the stability of K.F. algorithm is very difficult. 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 $HP(t)H^T + V_2$ ($p \times p$ matrix).

**Remark**

For LTI: $x(t+1) = Fx(t) + Gu(t)$ the stability check considers the eigenvalues of $F$.

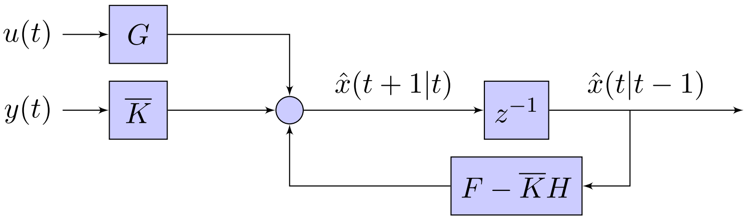
For LTV: $x(t+1) = F(t)x(t) + G(t)u(t)$, even if all the eigenvalues of $F(t)$ are strictly inside the unit circle at any time, the system is not guaranteed to be asymptotically stable. In practice it is, if the time-variations are *slow*, like in aging.

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

**Basic idea** If $P(t)$ converges to a constant value $\overline{P}$ (steady-state value of $P(t)$), then also $K(t)$ will converge to $\overline{K}$ (steady-state value of $K(t)$). Using $\overline{K}$ the K.F. becomes an LTI system.

Let's analyze the asymptotic stability of K.F. when $\overline{K}$ is used (assuming it exists).

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



If $\overline{K}$ exists, the K.F. is asymptotically stable if and only if all the eigenvalues of $F - \overline{K}H$ are strictly inside the unit circle.

**Remark**

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

K.F. can be asymptotically stable even if the system is unstable.

**Existance of $\overline{K}$**

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

$\overline{K}$ exists if $\overline{P}$ exists. We need to check the converge properties of D.R.E.

To find the equilibrium of a dynamical autonomous system:

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

D.R.E. is an autonomous discrete time system:

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

It's a non linear algebraic equation, known as Algebraic Riccati Equation (A.R.E.).

If a steady state $\overline{P}$ solution of D.R.E. does exists, it must be a solution of A.R.E. There remains 3 questions:

**Existence** Does A.R.E. have a semi-definite positive solution?

**Convergence** If exists, does the D.R.E. converges to $\overline{P}$?

**Stability** Is the corresponding $\overline{K}$ such that the K.F. is asymptotically stable?

Sergio
Savaresi
07/05/2020

To answer those questions we need two fundamental theorems (K.F. asymptotic theorems). They provide *sufficient* conditions only.

**First asymptotic theorem** Assumption: $V_{12} = 0$ and the system is asymptotically stable (all eigenvalues of $F$ are strictly inside the unit circle). Then:

- A.R.E. has one and only one semi-definite positive solution: $\overline{P} \geq 0$
- D.R.E. converges to $\overline{P}$, $\forall P_0 \geq 0$
- The corresponding $\overline{K}$ is s.t. the K.F. is asymptotically stable

**Recall**

**Observability** the pair $(F, H)$ is observable if and only if

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

**Controllability from the noise**

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

We need controllability from $v_1(t)$ (and don't care about $u(t)$).

$$x(t+1) = Fx(t) + v_1(t)$$

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:

$$R = \begin{bmatrix} \Gamma & F\Gamma & \cdots & F^{n-1}\Gamma \end{bmatrix}$$

**Second asymptotic theorem** Assumption: $V_{12} = 0$, $(F, H)$ is observable and $(F, \Gamma)$ is controllable. Then:

- A.R.E. has one and only one definite positive solution $\overline{P} > 0$
- D.R.E. converges to $\overline{P}$, $\forall P_0 \geq 0$
- The corresponding $\overline{K}$ is such that the K.F. is asymptotically stable

There theorems are very useful in practice because we can fully avoid the (very difficult) convergence analysis of D.R.E.

**Note** that the two theorems provide only sufficient conditions.

**Exercise**

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

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

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

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

**First step** Compute the D.R.E.

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

**Note** The second order terms must cancel out.

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

**Second step** Compute and solve the A.R.E.

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

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

$\overline{P} = 1$ is the only definite positive solution of A.R.E.

**Question** Does D.R.E. converges to $\overline{P} = 1$, $\forall P_0 \geq 0$?

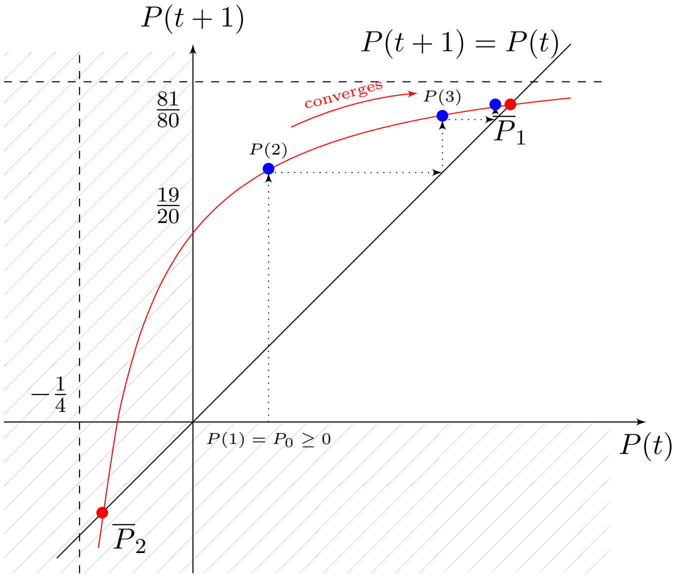
There are 2 methods for addressing this question:

- Direct analysis of D.R.E.
- Using asymptotic theorems

**First methods** Direct analysis of D.R.E.

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

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



By direct analysis/inspection of D.R.E. dynamics we can conclude that $\forall P_0 \geq 0$, D.R.E. 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

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

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.

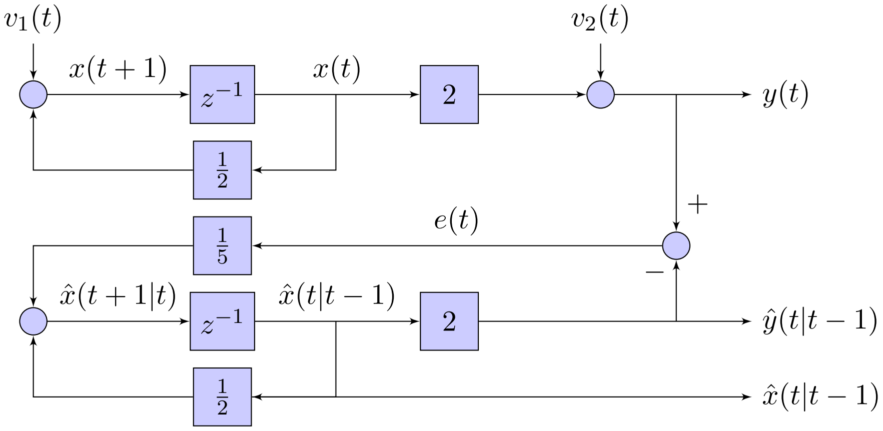
Both theorems are fulfilled, so A.R.E. has one and only one solution $\overline{P} > 0$, D.R.E. converges to $\overline{P}$, $\forall P_0 \geq 0$ and $\overline{K}$ makes the K.F. asymptotically stable.

**Third step** Compute $\overline{K}$

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

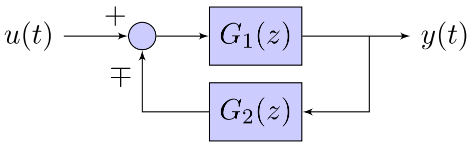
Double-check the asymptotical stability of K.F.

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



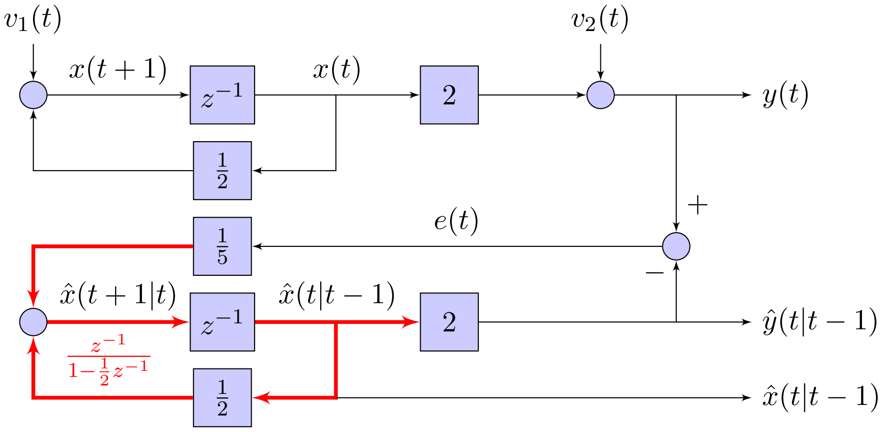
**Question** Find T.F. from $y(t)$ to $\hat{x}(t|t-1)$

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



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

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



Predictor of state:

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

Predictor of output:

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

Filter of state:

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

Sergio
Savaresi
11/05/2020

**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.

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

A model of $\eta(t)$ is given:

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

$\eta(t)$ is not a white noise, but an AR(1) stochastic model. We cannot apply K.F. formula to this system

We can proceed as follows

$$\begin{aligned} \eta(t) &= c\eta(t-1) + e(t) \\ \eta(t+1) &= c\eta(t) + e(t+1) \\ \eta(t+1) &= c\eta(t) + v(t) \qquad v(t) = e(t+1) \qquad v \sim WN(0,1) \qquad v \perp v_2 \end{aligned}$$

**Trick** Extension of the state vector.

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

$$\begin{cases} x_1(t+1) = ax_1(t) + x_2(t) \\ x_2(t+1) = cx_2(t) + v(t) \\ y(t) = bx_1(t) + v_2(t) \end{cases}$$

$$n = 2 \quad F \begin{bmatrix} a & 1 \\ 0 & c \end{bmatrix} \quad H = \begin{bmatrix} b & 0 \end{bmatrix} \quad v_1 = \begin{bmatrix} 0 \\ v(t) \end{bmatrix} \quad V_1 = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \quad v_2 \sim WN(0,1) \quad V_{12} = 0$$

We can apply K.F. to this system.

### 3.4.1 Extension to Non-Linear systems

Consider a system with non-linear dynamics:

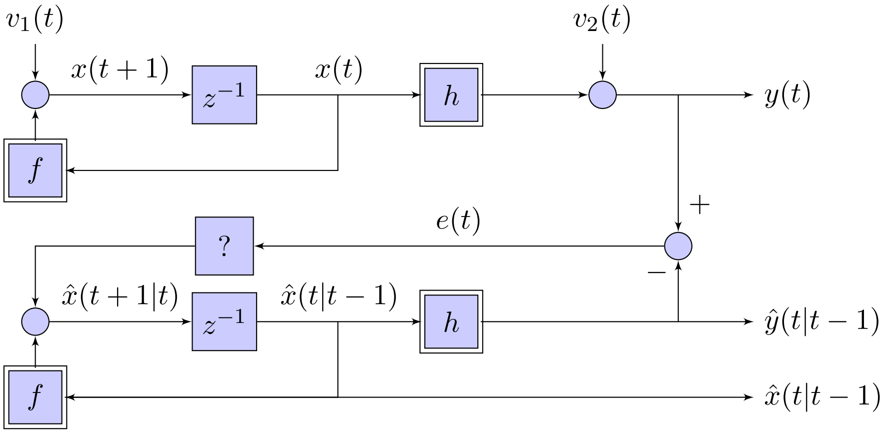
$$S : \begin{cases} x(t+1) = f(x(t), u(t)) + v_1(t) \\ y(t) = h(x(t)) + v_2(t) \end{cases}$$

Where $f$ and $h$ are non-linear functions of $x(t)$ and $u(t)$ (smoothness class $C^1$ or higher).

**Example**

$$S : \begin{cases} x(t+1) = \frac{1}{2}x^5(t) + u^3(t) + v_1(t) \\ y(t) = e^{x(t)} + v_2(t) \end{cases}$$

How can we design a Kalman Filter in this case?



For the gain block of K.F. we have 2 different types of solutions:

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

The second solution is less intuitive but is the most effective: we can reuse in full K.F. formulas with small adjustments. K.F. can be applied to LTV systems.

In practice Extended Kalman Filter idea is to make a time-varying linear local approximation of a non-linear time-invariant system.

$K(t)$ 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 D.R.E.

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

- Equations of $K$ and DRE are the usual formulas of K.F.
- The only difference is $F(t)$ and $H(t)$ are computed as follows

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

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

Procedure to implement EKF at time $t$:

- Take last available state prediction $\hat{x}(t|t-1)$
- Using $\hat{x}(t|t-1)$, compute $F(t)$ and $H(t)$
- Compute $K(t)$ and update D.R.E. equations
- Compute $\hat{x}(t+1|t)$

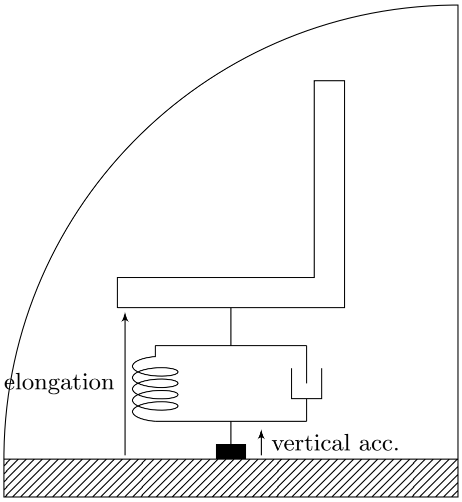
**Remark**

- EKF is very powerful since can be applied to non-linear systems
- Obviously EKF does not have a steady-state asymptotic solution
- Main problems with EKF (same of L.T.V. K.F.):
  - Very difficult (almost impossible) to have a theoretical guarantee of E.K.F. stability (in practice extensive empirical testing).
  - Computational load (at each time $F(t)$, $H(t)$, $K(t)$ and $P(t)$ must be computed run-time).

EKF is largely used today with some limitations in:

- safety-critical applications
- mission-critical applications

**Exercise: K.F. full procedure**



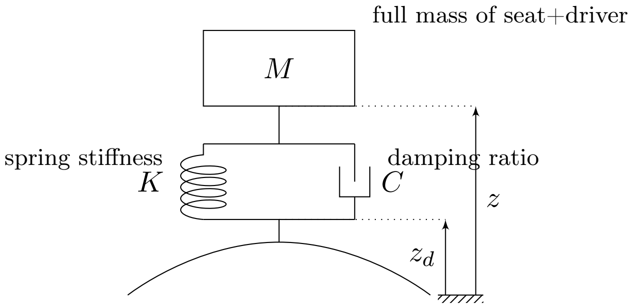
Suspended seat in the cabin of an off-highway vehicle (agr. tractor, earth-moving machine, etc. . . ).

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.

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

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

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

Next is discretization (we use digital systems), choice of sampling time $\Delta$ (for this application can be 5ms).

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

Example for the first equation:

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

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

This is normal state-space for in discrete time:

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

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

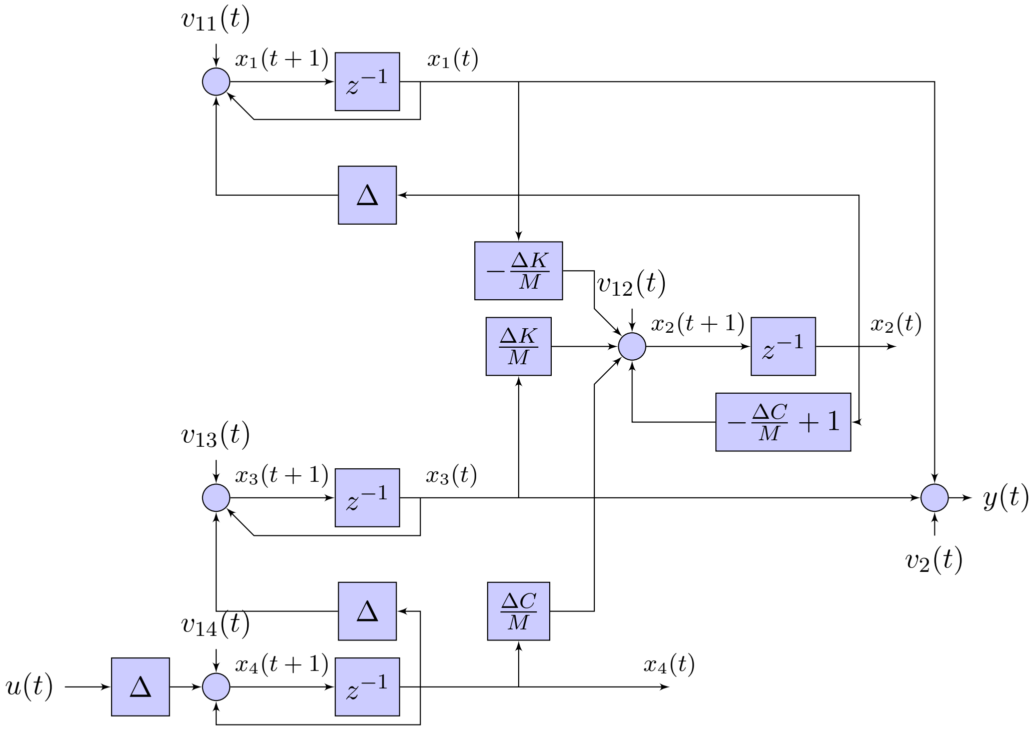
Noise on the state equations:

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

Assumptions:

- All white-noises
- All uncorrelated

$V_2$ can be estimated by datasheet of elongation sensor, $\lambda_4$ can be estimated by datasheet of accelerometer. We expect $\lambda_1$, $\lambda_2$ and $\lambda_3$ 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.

K.F. at this point can be applied. Theorem 1 and 2 are valid, we can directly jump to A.R.E. solution.

Sergio
Savaresi
12/05/2020

### 3.4.2 Direct optimization of gain $K$

$$S : \begin{cases} x(t+1) = 2x(t) \\ y(t) = x(t) + v(t) \quad v \sim WN(0,1) \end{cases}$$

**Problem** Compute the steady-state predictor of state $\hat{x}(t+1|t)$.

We use 2 different methods:

1. Direct optimization
2. K.F. theory

**Remark**

The system is not stable and the state equation is noise-free.

**First mehtod** Direct solution

Let's start from the standard observer structure

$$\begin{cases} \hat{x}(t+1|t) = 2\hat{x}(t|t-1) + K(y(t) - \hat{y}(t|t-1)) \\ \hat{y}(t|t-1) = \hat{x}(t|t-1) \end{cases}$$

Find the optimal $K$ by minimizing the variance of the state prediction error var $[x(t) - \hat{x}(t|t-1)]$.

Write the state prediction error expression:

$$\begin{aligned} x(t+1) - \hat{x}(t+1|t) &= 2x(t) - [2\hat{x}(t|t-1) + K(y(t) - \hat{y}(t|t-1))] = \\ &= 2x(t) - 2\hat{x}(t|t-1) - K(x(t) + v(t) - \hat{x}(t|t-1)) = \\ &= (2-K)(x(t) - \hat{x}(t|t-1)) - Kv(t) \end{aligned}$$

**Definition** $\eta(t) = x(t) - \hat{x}(t|t-1)$

The dynamic equation of the state prediction error is:

$$\eta(t+1) = (2-K)\eta(t) - Kv(t) \qquad v \sim WN(0,1)$$

It is an AR(1) process, in canonical form:

$$\eta(t) = \frac{1}{1-(2-K)z^{-1}}e(t) \qquad e(t) = -Kv(t) \qquad e \sim WN(0, K^2)$$

It's easy to find the variance of $\eta(t)$:

$$\gamma_\eta(0) = \text{var}\left[\eta(t)\right] = \frac{K^2}{1-(2-K)^2}$$

By minimizing this function with respect to $K$:

$$\frac{\partial \text{var}\left[\eta(t)\right]}{\partial K} = 0 \quad \Rightarrow \quad \begin{cases} K_1 = 0 & \text{(no need of feedback correction)} \\ K_2 = \frac{3}{2} \end{cases}$$

Both are acceptable solutions.

**Second method** Using K.F. theory

From the system $S$: $F = 2$, $H = 1$, $V_1 = 0$ so $\Gamma = 0$, $V_2 = 1$, $V_{12} = 0$.

Theorem 1 requires that $V_{12} = 0$ (ok) and $F$ is asymptotically stable (not ok).

Theorem 2 requires that $V_{12} = 0$ (ok), $(F, \Gamma)$ not reachable (not ok), $(F, H)$ is observable (ok).

We cannot skip the analysis of D.R.E.

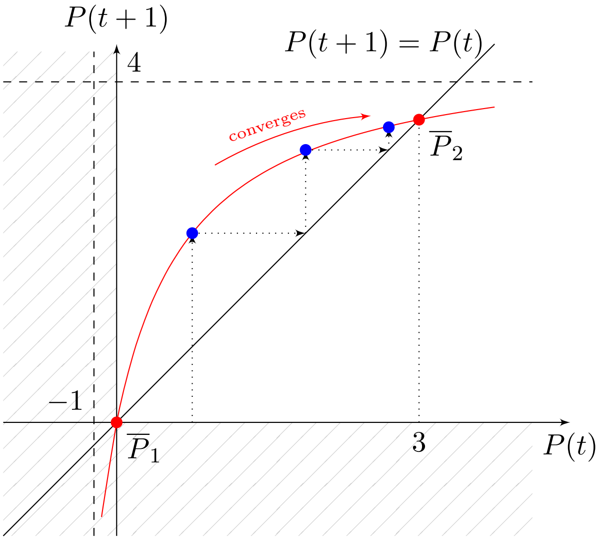
$$\text{D.R.E.} = 4P(t) - \frac{(2P(t))^2}{P(t)+1} \dots$$

$$P(t+1) = \frac{4P(t)}{P(t)+1}$$

Next, find the corresponding the A.R.E. solutions.

$$\overline{P} = \frac{4\overline{P}}{\overline{P}+1} \quad \Rightarrow \quad \begin{cases} \overline{P}_1 = 0 \\ \overline{P}_2 = 3 \end{cases} \quad \Rightarrow \quad \begin{cases} K_1 = 0 \\ K_2 = \frac{3}{2} \end{cases}$$

We need to make the D.R.E. convergence analysis.

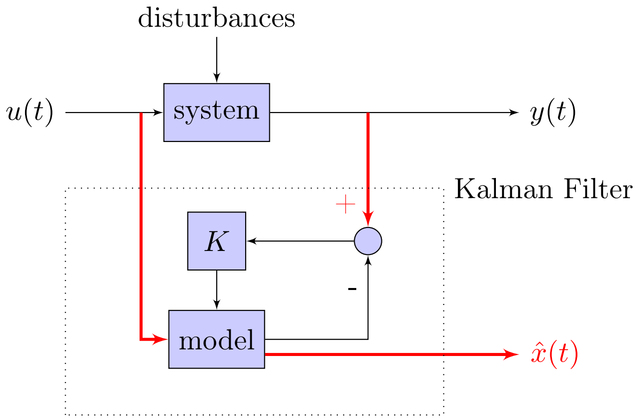


If we start from $P_0 = 0$ (no uncertainty in the knowledge of $x(1)$) it converges into $\overline{P}_1 = 0$, the state equation is noise-free and we don't need feedback and the prediction is perfect (open-loop solution). This is a feasible but ideal situation.

The standard solution (no perfect initial condition) is when $K = \frac{3}{2}$ (real closed-loop solution).

# 4 Software-sensing with Black-Box Methods

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



Main features of this approach:

- A (white-box/physical) model must be available.
- 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.

In this chapter we see them focusing on the architecture (we do not need new algorithms, just use something we have already studied).

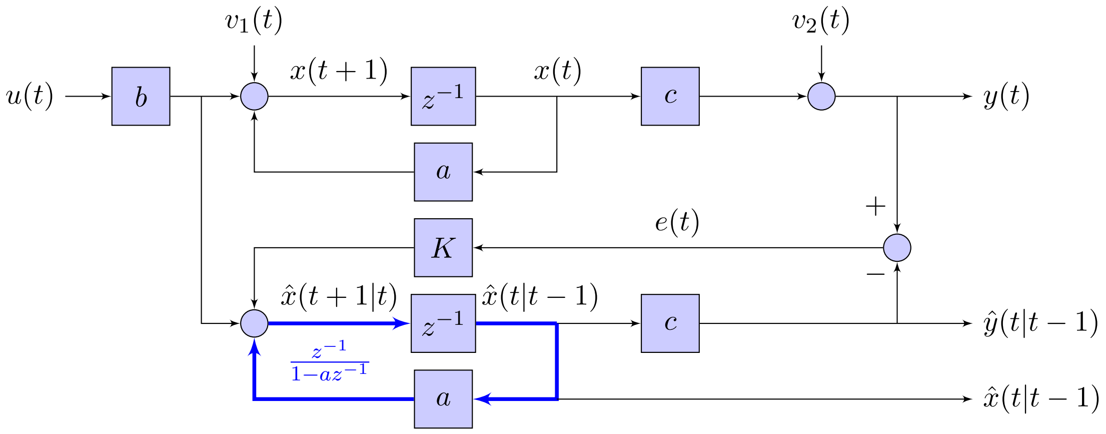
## 4.1 Linear Time Invariant Systems

Let's consider a simplified case (SISO system with one state) to understand the approach.

$$S : \begin{cases} x(t+1) = ax(t) + bu(t) + v_1(t) \\ y(t) = cx(t) + v_2(t) \end{cases}$$

**Problem** Estimation of $\hat{x}(t)$ from measured signals $u(t)$ and $y(t)$.

We can start from the observer K.F. architecture:



Let's find the relationship between $u(t) \to \hat{x}(t)$ and $y(t) \to \hat{x}(t)$.

$$\begin{aligned} \hat{x}(t) &= \frac{b\frac{z^{-1}}{1-az^{-1}}}{1+Kc\frac{z^{-1}}{1-az^{-1}}}u(t) + \frac{K\frac{z^{-1}}{1-az^{-1}}}{1+Kc\frac{z^{-1}}{1-az^{-1}}}y(t) \\ &= \frac{b}{1+(Kc-a)z^{-1}}u(t-1) + \frac{K}{1+(Kc-a)z^{-1}}y(t-1) \end{aligned}$$

Sergio
Savaresi
14/05/2020

We can make a Black-Box estimation of these two transfer functions from data.

K.F. is a sophisticated way to build those T.F. from a white-box model. We can estimate these functions directly from data.

We can adopt a black-box approach to estimate these T.F.:

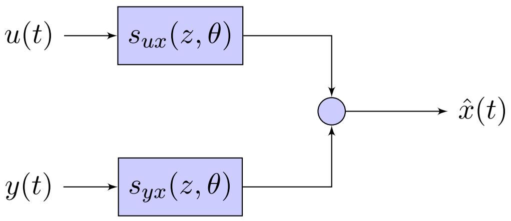
### Dataset for training

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

In the supervised training approach we need measurements of the state to be estimated (physical sensor for $x(t)$ only for design/training of the software-sensor).

We can use 4SID for direct non-parametric identification of $(u, y) \to x$ dynamics, or we can use classic parametric system identification approach:

### Model selection



### Performance index

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

### Optimization

$$\hat{\theta}_N = \arg\min_{\theta} J_N(\theta)$$

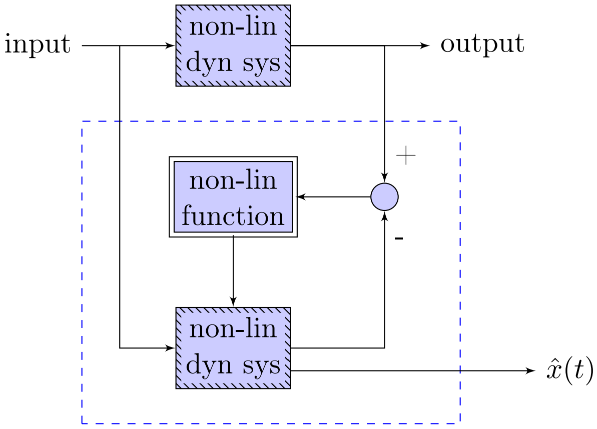
We obtain $S_{ux}(z, \hat{\theta}_N)$ and $S_{yx}(z, \hat{\theta}_N)$, the black-box software sensor.

## 4.2 Comparison between K.F. and B.B. software sensing

| | **K.F.** | **B.B.** |
|---|---|---|
| Need of (WB) physical model of the system | Yes | No |
| Need a training set | No | Yes |
| Interpretability of the result | Yes | No |
| Easy retuning for a similar (different) system | Yes | No |
| Accuracy of the estimation | Good | Very Good |
| Can be used also in case of un-measurable states | Yes | No |

## 4.3 Non-linear Systems

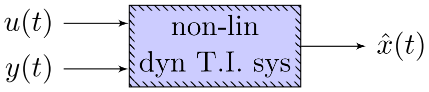
When the system is non-linear the problem becomes more complicated. Let's take inspiration from E.K.F.



**Remark**

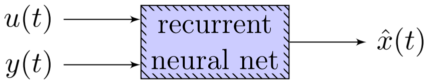
In K.F. the E.K.F. extension uses the trick of a time-varying linear gain $K(t)$ but the obvious choice is a non-linear gain (static nonlinear function).

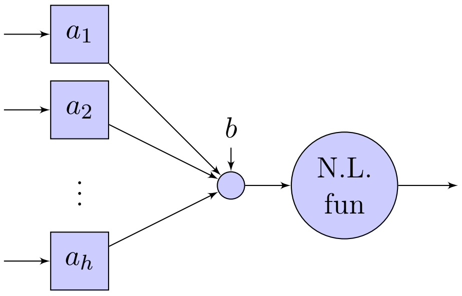
The content of the box is:



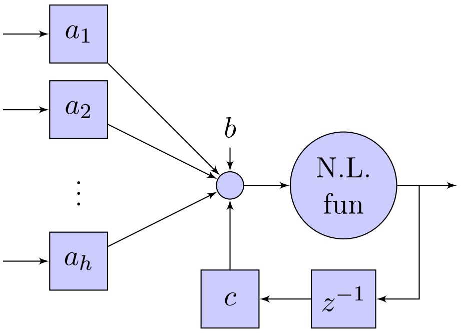
The problem is again the B.B. identification of a non-linear dynamic system, starting from a measured training dataset.

**Architecture #1** Use a recurrent neural network





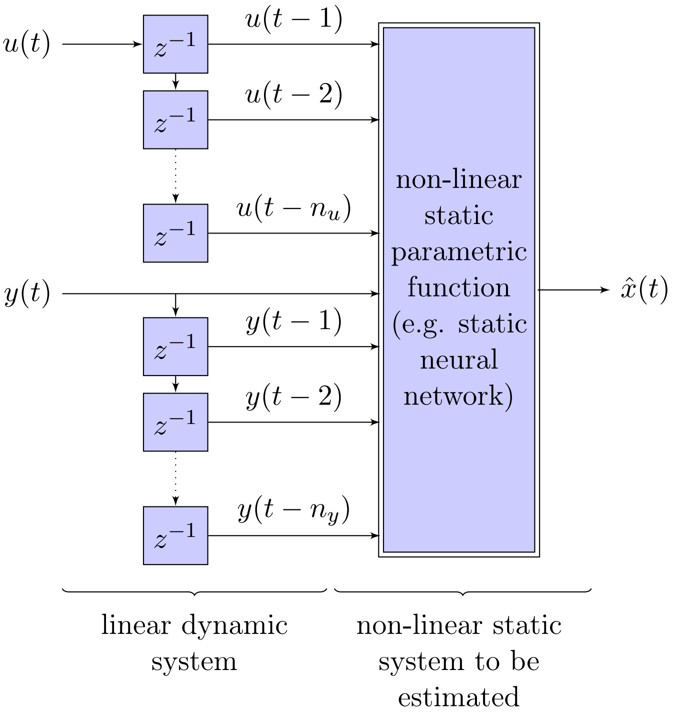
Static function of a neuron



Upgrade to a dynamic neuron

Most general approach but practically seldom used: major issues of stability and convergence of training.

**Architecture #2** Split the system into a static non-linear system and linear dynamics (F.I.R. architecture)


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

**Remark**

Notice that in principle $\hat{x}(t)$ can depend on $y(t)$ whereas we know $\hat{x}(t)$ can only depend on $u(t-1)$ and past values.

In case of a MIMO system with

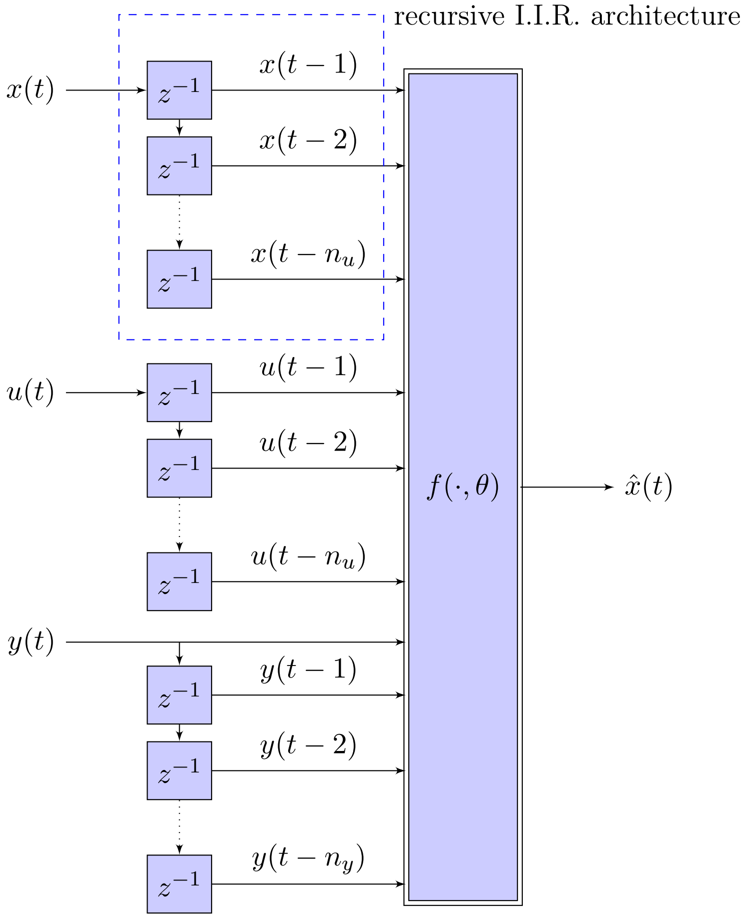
$$m \text{ inputs: } u(t) = \begin{bmatrix} u_1(t) \\ \vdots \\ u_m(t) \end{bmatrix} \quad p \text{ outputs: } y(t) = \begin{bmatrix} y_1(t) \\ \vdots \\ y_p(t) \end{bmatrix} \quad n \text{ states: } x(t) = \begin{bmatrix} x_1(t) \\ \vdots \\ x_n(t) \end{bmatrix}$$

The estimation problem is the search of the optimal parameter vector $\theta$ for the function

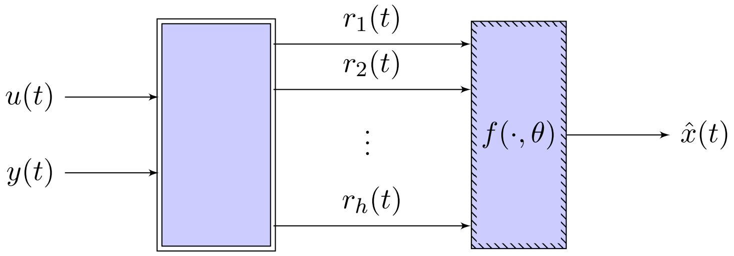
$$f(\cdot, \theta) : \mathbb{R}^{m \times n_u + p \times (n_y+1)} \rightarrow \mathbb{R}^n$$

> The estimation of this function $f(\cdot,\theta)$ is much simpler than the estimation of a recurrent neural network. Moreover the stability is guaranteed (the system is F.I.R.).

**Architecture #3** Static non-linear function plus linear dynamics but with a I.I.R. scheme



**Architecture #4** Separation of system dynamics and a static non-linear system using regressors built from physical knowledge



The system (can be dynamic and non-linear) that builds the regressors (signals $r_1(t)$, $r_2(t)$, $\dots$) from the physical sensors $u(t)$ and $y(t)$ using physical knowledge of the system.

The idea is to facilitate the job of $f(\cdot, \theta)$ by presenting at its input a smaller and more meaningful set of signals (regressors).

**Exercise: Continue from the last one**

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:

$$\begin{aligned} c(t) &: \{c(1), c(2), \cdots, c(N)\} \\ z(t) - z_d(t) &: \{z(1) - z_d(1), z(2) - z_d(2), \cdots, z(N) - z_d(N)\} \\ \ddot{z}(t) &: \{\ddot{z}(1), \ddot{z}(2), \cdots, \ddot{z}(N)\} \\ \dot{z}(t) &: \{\dot{z}(1), \dot{z}(2), \cdots, \dot{z}(N)\} \text{ (just for training)} \end{aligned}$$

Back to the main equation:

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

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

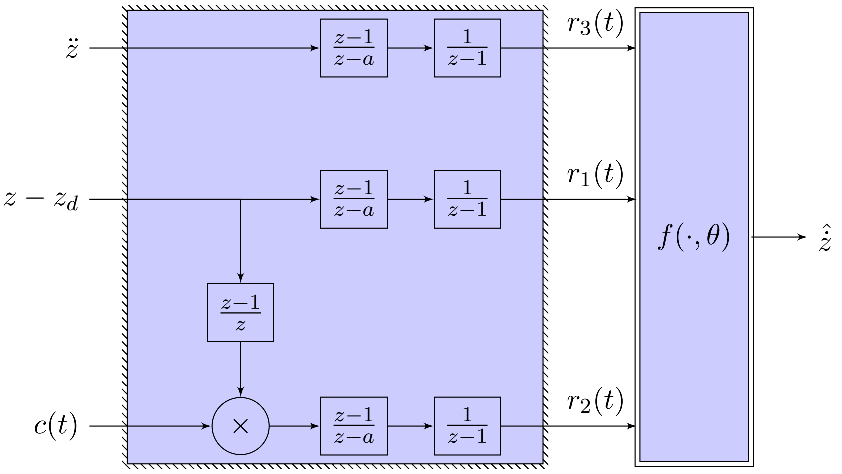
We also consider this equation

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

$r_1(t)$ 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

Sergio Savaresi 18/05/2020

**Conclusions** In case of black-box software 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.

# 5 Gray-box System Identification

We'll see two approaches

- Using Kalman Filter
- Using Simulation error method

## 5.1 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 K.F.).

**Problem definition** We have a model, typically built as a white-box model using first principles:

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

$f$ and $h$ are linear or non-linear functions, depending on some unknown parameter $\theta$ (with a physical meaning, e.g. mass, resistance, friction, …).

The problem is to estimate $\hat{\theta}$.

K.F. solves this problem by transforming the unknown parameters in extended states: K.F. makes the simultaneous estimation of $\hat{x}(t|t)$ (classic Kalman Filter problem) and $\hat{\theta}(t)$ (parameter identification problem).

**Trick** State extension

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

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

The new equation we created

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

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

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

We need a *fictitious* noise in order to force Kalman Filter to find the right value of $\theta$ (if no noise in this equation K.F. probably would stay fixed on the initial condition). We tell K.F. to do not rely on initial conditions.

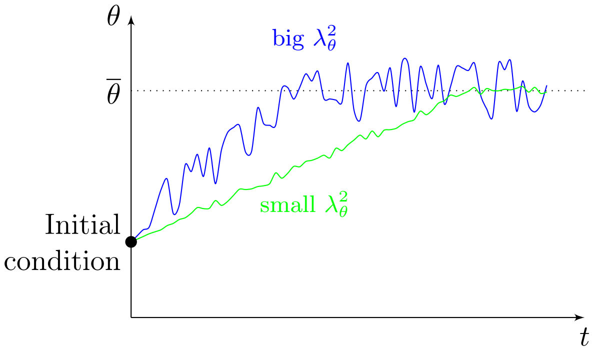
Notice that this equation is not of an asymptotic stable system but a simply-stable system. It's not a problem because K.F. can deal with non-asymptotically stable systems.

**Design choice** The choice of the covariance matrix of $v_\theta(t) \sim WN(0, V_\theta)$.

We make the empirical assumption that $v_1 \perp v_\theta$ and $v_2 \perp v_\theta$ (there is no reason for $v_\theta$ to be correlated with $v_1$ and $v_2$).

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

It is a $n_\theta \times n_\theta$ and usually it is assumed that $\lambda_{1\theta}^2 = \lambda_{2\theta}^2 = \cdots = \lambda_{n_\theta\theta}^2$. We assume that $v_\theta(t)$ is a set of independent W.N. all with the same variance $\lambda_\theta^2$ (tuned empirically).



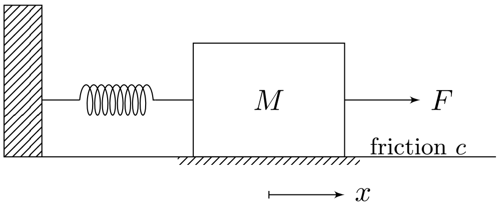
Influence of choice of $\lambda_\theta^2$

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

$\lambda_\theta^2$ is selected according to the best compromise for your specific application.

**Notice** 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).

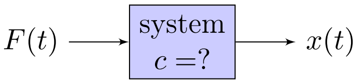
**Example**



**Input** $F(t)$

**Output** Position $x(t)$ (measured)

**Parameters** $K$ and $M$ are known (measured), $c$ is unknown



**Problem** Estimate $c$ with a K.F.

Using K.F. we do not need a training dataset.

**Step 1** Model the system

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

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

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

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

**Step 3** State extension

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

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

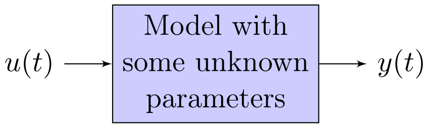
The system is ready for K.F. application: we get at the same time $\hat{x}(t)$ and $\hat{c}(t)$.

Notice that we need Extended Kalman Filter: even if the original system was linear, state extension moved to a non-linear system.

Sergio
Savaresi
19/05/2020

## 5.2 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).



**Step 1** Collect data from an experiment

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

**Step 2** Define model structure

$$y(t) = \mathcal{M}(u(t), \overline{\theta}, \theta)$$

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

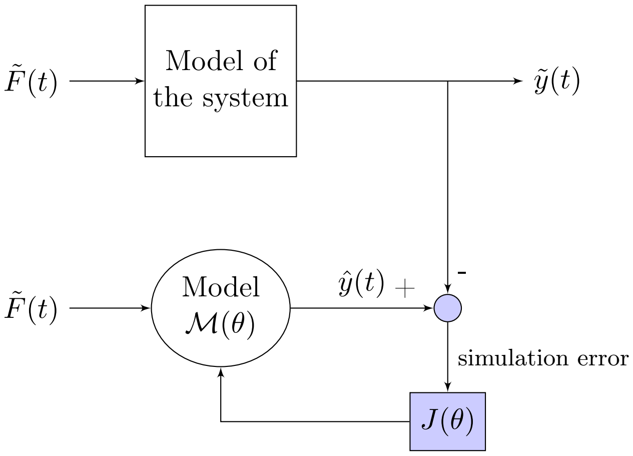
**Step 3** Performance index definition

$$J_N(\theta) = \frac{1}{N} \sum_{t=1}^{N} \left(\tilde{y}(t) - \mathcal{M}(\tilde{u}(t), \overline{\theta}, \theta)\right)^2$$

**Step 4** Optimization

$$\hat{\theta}_N = \arg\min_{\theta} J_N(\theta)$$

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



Can S.E.M. be applied also to B.B. methods?

**Example**

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

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

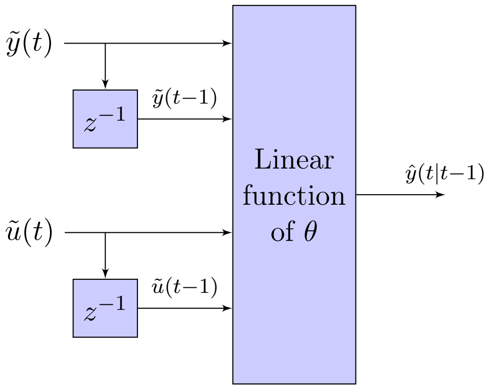
In time domain $y(t) = -a_1 y(t-1) - a_2 y(t-2) + b_0 u(t-1) + b_1 u(t-2)$.

Using P.E.M.

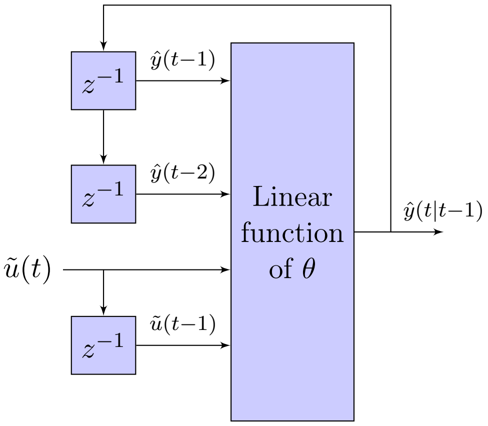
$$\hat{y}(t|t-1) = -a_1 \hat{y}(t-1) - a_2 \hat{y}(t-2) + b_0 \hat{u}(t-1) + b_1 \hat{u}(t-2)$$

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

Notice that it's a quadratic formula.



P.E.M.



S.E.M.

Using S.E.M.

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

$$\begin{aligned} J_N(\theta) &= \frac{1}{N}\sum_{t=1}^{N} (\tilde{y}(t) - \hat{y}(t|t-1,\theta))^2 \\ &= \frac{1}{N}\sum_{t=1}^{N} (\tilde{y}(t) + a_1\hat{y}(t-1) + a_2\hat{y}(t-2) - b_0\tilde{u}(t-1) - b_1\tilde{u}(t-2))^2 \end{aligned}$$

Notice that it's non-linear with respect to $\theta$.

P.E.M. approach looks much better, but do not forget the noise! P.E.M. 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 ARX models:

$$\begin{aligned} y(t) &= \frac{b_0 + b_1 z^{-1}}{1 + a_1 z^{-1} + a_2 z^{-2}} u(t-1) + \frac{1}{1 + a_1 z^{-1} + a_2 z^{-2}} e(t) \\ \hat{y}(t|t-1) &= b_0 u(t-1) + b_1 u(t-2) - a_1 y(t-1) - a_2 y(t-2) \end{aligned}$$

If we use ARMAX models the numerator of the T.F. for $e(t)$ is $1 + c_1 z^{-1} + \ldots + c_m z^{-m}$, in this case $J_N(\theta)$ is non-linear. This leads to the same complexity of S.E.M.

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

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

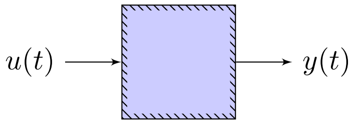
If $\Delta$ is very small the difference between $\tilde{y}(t)$ and $\tilde{y}(t-1)$ becomes very small. The effect is that the P.E.M. optimization ends to provide this *trivial* solution:

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

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 Conclusion

Summary of system identification methods for I/O systems



- Collect a dataset for training (if needed)
- Choose a model domain (linear static/non-linear static/linear dynamic/non-linear dynamic), using gray-box or black-box
- Estimation method: constructive (4SID), parametric (P.E.M. or S.E.M.) or filtering (state extension of K.F.)

Better black-box for system identification and software-sensing or white box?

It depends on the goals and type of applications.

- Black box is very general and very flexible, make maximum use of data and no or little need of domain knowhow
- White box is very useful when you are the system-designer (not only the control algorithm designer), can provide more insight in the system.
- Gray box can sometimes be obtained by hybrid systems (part is black-box and part is white-box).

# 6 Minimum Variance Control

Sergio Savaresi 25/05/2020

Design and analysis of feedback systems. It's not system identification nor software sensing.

- 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).

**Setup of the problem** Consider a generic ARMAX model:

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

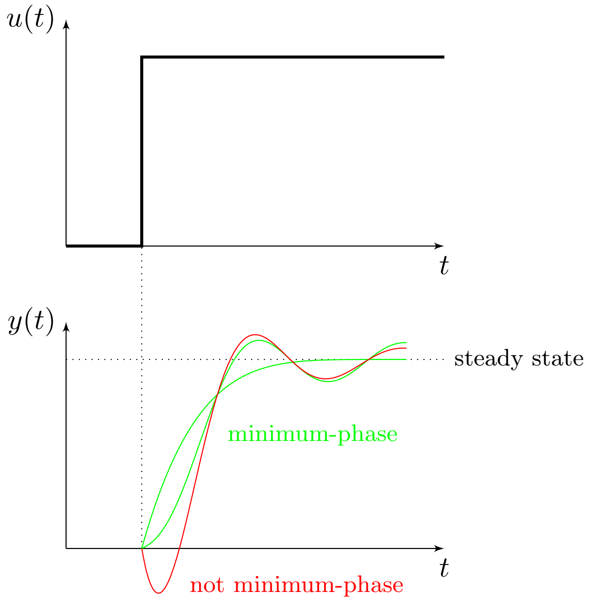
$$\begin{aligned} B(z) &= b_0 + b_1 z^{-1} + \cdots + b_p z^{-p} \\ A(z) &= 1 + a_1 z^{-1} + \cdots + a_m z^{-m} \\ C(z) &= 1 + c_1 z^{-1} + \cdots + c_n z^{-n} \end{aligned}$$

Assumptions:

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

**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.



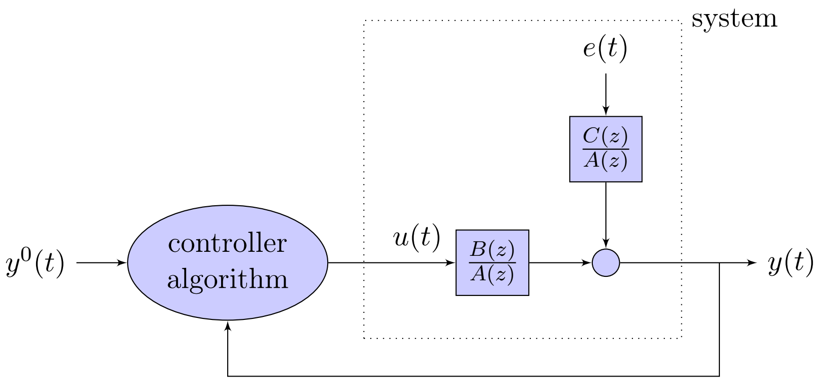
At the very beginning the response of a non-minimum phase system goes to the opposite direction w.r.t. the final value.

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).

The problem we wish to solve is optimal tracking of the desired behavior of output:



In a more formal way MVC tries to minimize this performance index:

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

It's the variance of the tracking error, that's why it's called Minimum Variance Control.

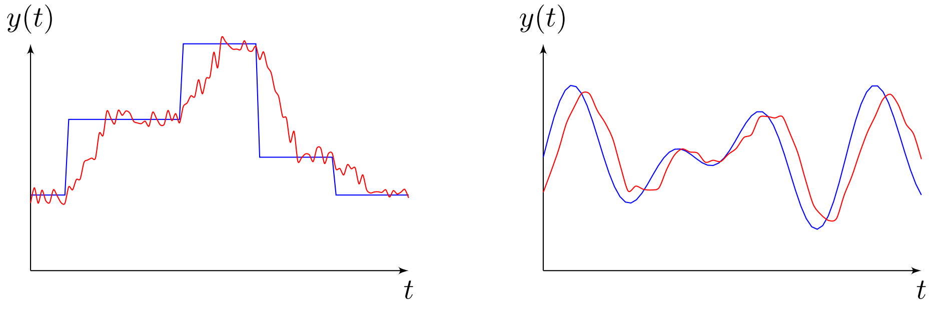
Some additional (small) technical assumptions:

- $y^0(t)$ and $e(t)$ are not correlated (usually fulfilled)
- We assume that $y^0(t)$ is known only up to time $t$ (present time), we have no preview of the future desired $y^0(t)$ ($y^0(t)$ is totally unpredictable).

**Remark**

There are 2 sub-classes of control problems:

- When $y^0(t)$ is constant or step-wise (regulation problem)
- When $y^0(t)$ is varying (tracking problem)



Regulation problem    Tracking problem

Bottom-up way of presenting M.V.C.

### Simplified problem #1

$$S: y(t) = ay(t-1) + b_0u(t-1) + b_1u(t-2) \qquad y(t) = \frac{b_0 + b_1z^{-1}}{1 - az^{-1}}u(t-1)$$

We assume that $y^0(t) = \overline{y}^0$ (regulation problem) and the system is noise-free.

- $b_0 \neq 0$
- Root of numerator must be inside the unit circle

To design the minimum variance controller we must minimize the performance index:

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

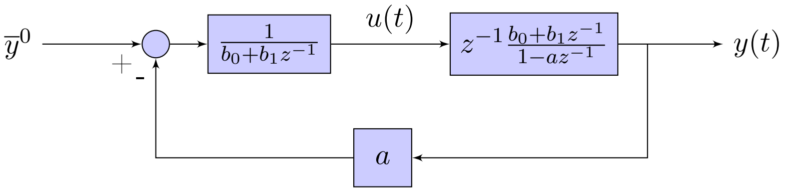
There is no noise so we can remove the expected value

$$\begin{aligned} J &= \left(y(t) - y^0(t)\right)^2 = \left(y(t) - \overline{y}^0\right)^2 = \left(ay(t-1) + b_0u(t-1) + b_1u(t-2) - \overline{y}^0\right)^2 = \\ &= \left(ay(t) + b_0u(t) + b_1u(t-1) - \overline{y}^0\right)^2 \\ \frac{\partial J}{\partial u(t)} &= 2\left(ay(t) + b_0u(t) + b_1u(t-1) - \overline{y}^0\right)(b_0) \end{aligned}$$

Why the derivative is just $b_0$? We are at present time $t$ and at time $t$ the control algorithm must take a decision on the value of $u(t)$. At time $t$, $y(t)$, $y(t-1)$, $\dots$, $u(t-1)$, $u(t-2)$, $\dots$ are no longer variables but numbers.

$$\frac{\partial J}{\partial u(t)} = 0 \qquad ay(t) + b_0u(t) + b_1u(t-1) - \overline{y}^0 = 0$$

$$u(t) = \left(\overline{y}^0 - ay(t)\right)\frac{1}{b_0 + b_1z^{-1}}$$


$\overline{y}^0$
+
-
$\frac{1}{b_0+b_1z^{-1}}$
$u(t)$
$z^{-1}\frac{b_0+b_1z^{-1}}{1-az^{-1}}$
$y(t)$
$a$

### Simplified problem #2

$$S: y(t) = ay(t-1) + b_0u(t-1) + b_1u(t-2) + e(t) \qquad e(t) \sim WN(0, \lambda^2)$$

The reference variable is $y^0(t)$ (tracking problem).

The performance index is

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

The fundamental trick to solve this problem is to re-write $y(t)$ as:

$$y(t) = \hat{y}(t|t-1) + \epsilon(t)$$

Since $k = 1$ we know that $\epsilon(t) = e(t)$, so $y(t) = \hat{y}(t|t-1) + e(t)$.

$$\begin{aligned} J &= E\left[\left(\hat{y}(t|t-1) + e(t) - y^0(t)\right)^2\right] \\ &= E\left[\left((\hat{y}(t|t-1) - y^0(t)) + e(t)\right)^2\right] \\ &= E\left[\left(\hat{y}(t|t-1) - y^0(t)\right)^2\right] + E\left[e(t)^2\right] + \cancel{2E\left[e(t)\left(\hat{y}(t|t-1) - y^0(t)\right)\right]} \end{aligned}$$

Notice that

$$\arg\min_{u(t)} \left\{E\left[\left(\hat{y}(t|t-1) - y^0(t)\right)^2\right] + \lambda^2\right\} = \arg\min_{u(t)} \left\{E\left[\left(\hat{y}(t|t-1) - y^0(t)\right)^2\right]\right\}$$

The best result is when $\hat{y}(t|t-1) = y^0(t)$, we can force this relationship.

Now we must compute the 1-step predictor of the system:

$$S : y(t) = \frac{b_0 + b_1 z^{-1}}{1 - az^{-1}} u(t-1) + \frac{1}{1 - az^{-1}} e(t)$$

Note that this is an $ARMAX(1, 0, 1+1) = ARX(1, 2)$.

$$k = 1 \qquad B(z) = b_0 + b_1 z^{-1} \qquad A(z) = 1 - az^{-1} \qquad C(z) = 1$$

General solution for 1-step prediction of ARMAX:

$$\hat{y}(t|t-1) = \frac{B(z)}{C(z)} u(t-1) + \frac{C(z) - A(z)}{C(z)} y(t)$$

If we apply this formula we obtain:

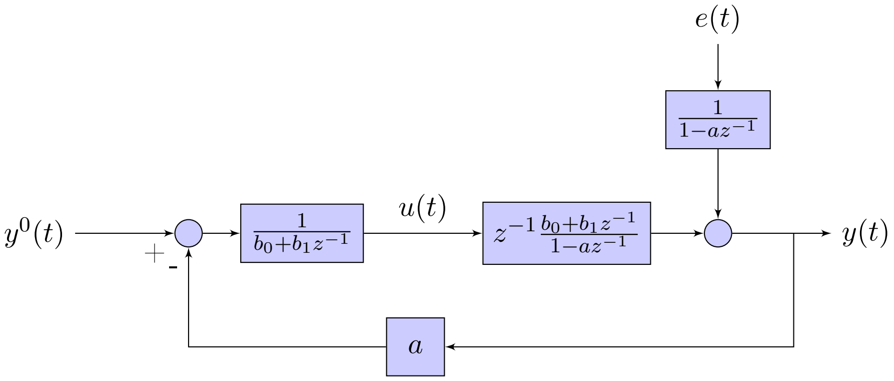
$$\hat{y}(t|t-1) = \frac{b_0 + b_1 z^{-1}}{1} u(t-1) + \frac{1 - 1 + az^{-1}}{1} y(t) = (b_0 + b_1 z^{-1}) u(t-1) + ay(t-1)$$

Now we can impose that $\hat{y}(t|t-1) = y^0(t)$

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

But we don't have $y^0(t+1)$, so we use $y^0(t)$.

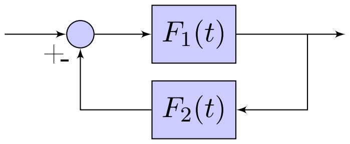
$$u(t) = \left(y^0(t) - ay(t)\right) \frac{1}{b_0 + b_1 z^{-1}}$$


$e(t)$
$\frac{1}{1-az^{-1}}$
$y^0(t)$
+
-
$\frac{1}{b_0+b_1z^{-1}}$
$u(t)$
$z^{-1}\frac{b_0+b_1z^{-1}}{1-az^{-1}}$
$y(t)$
$a$

Sergio
Savaresi
27/05/2020

## Remark

For stability let's recall a result of feedback system:



To check the closed-loop stability:

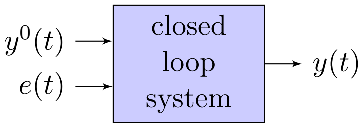
- compute the *loop-function* $L(z) = F_1(z)F_2(z)$ (**do not simplify!**)
- Build the *characteristic polynomial* $\chi(z) = L_N(z) + L_D(z)$ (sum of numerator and denominator)
- Find the roots of $\chi(z)$, closed loop system is asymptotically stable iff all the roots of $\chi(z)$ are strictly inside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

$$L(z) = \frac{1}{b_0 + b_1z^{-1}} \cdot \frac{z^{-1}(b_0 + b_1z^{-1})}{1 - az^{-1}} \cdot a$$

$$\chi(z) = az^{-1}(b_0 + b_1z^{-1}) + (1 - az^{-1})(b_0 + b_1z^{-1}) = b_0 + b_1z^{-1} = B(z)$$

The closed loop system is asymptotically stable thanks to the minimum phase assumption.

Performance analysis



Since the system is L.T.I. we can use the superposition principle:

$$y(t) = \underbrace{F_{y^0y}(z)y^0(t)}_{\text{TF from } y^0 \text{ to } y} + \underbrace{F_{ey}(z)e(t)}_{\text{TF from } e \text{ to } y}$$

Let's compute these transfer functions:

$$\begin{aligned}
F_{y^0y} &= \frac{\frac{1}{b_0+b_1z^{-1}}\frac{z^{-1}(b_0+b_1z^{-1})}{1-az^{-1}}}{1+\frac{1}{b_0+b_1z^{-1}}\frac{z^{-1}(b_0+b_1z^{-1})}{1-az^{-1}}a} = z^{-1} \\
F_{ey} &= \frac{\frac{1}{1-az^{-1}}}{1+\frac{1}{b_0+b_1z^{-1}}\frac{z^{-1}(b_0+b_1z^{-1})}{1-az^{-1}}a} = 1
\end{aligned}$$

Notice that that the closed loop system has a very simple closed-loop behavior:

$$y(t) = z^{-1}y^0(t) + e(t) = y^0(t-1) + e(t)$$

$y(t)$ follows exactly $y^0(t)$ (but with a 1-step delay), disturbed by noise $e(t)$. This is the best possible solution.

## 6.1 General solution

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

With the assumptions:

- $b_0 \neq 0$
- $B(z)$ has all the roots strictly inside the unit circle (minimum phase)
- $\frac{C(z)}{A(z)}$ is canonical representation
- $y^0(t) \perp e(t)$
- $y^0(t)$ is unpredictable

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

The trick is to rewrite $y(t) = \hat{y}(t|t-k) + \epsilon(t)$.

$$\begin{aligned}
J &= E\left[\left(\hat{y}(t|t-k) + \epsilon(t) - y^0(t)\right)^2\right] = E\left[\left((\hat{y}(t|t-k) - y^0(t)) + \epsilon(t)\right)^2\right] \\
&= E\left[\left(\hat{y}(t|t-k) - y^0(t)\right)^2\right] + E\left[\epsilon(t)^2\right] + \cancel{2E\left[\epsilon(t)\left(\hat{y}(t|t-k) - y^0(t)\right)\right]} \\
&= E\left[(\hat{y}(t|t-k) - y^0(t))^2\right] + \text{constant w.r.t. } u(t)
\end{aligned}$$

General formula for ARMAX predictor:

$$\frac{C(z)}{A(z)} = \underbrace{E(z)}_{\text{solution}} + \underbrace{\frac{\tilde{R}(z)z^{-n}}{A(z)}}_{\text{residual}} \qquad \text{(after $n$ steps of long division)}$$

$$\hat{y}(t|t-k) = \frac{B(z)E(z)}{C(z)}u(t-k) + \frac{\tilde{R}(z)}{C(z)}y(t-k)$$

Shift ahead $k$-steps:

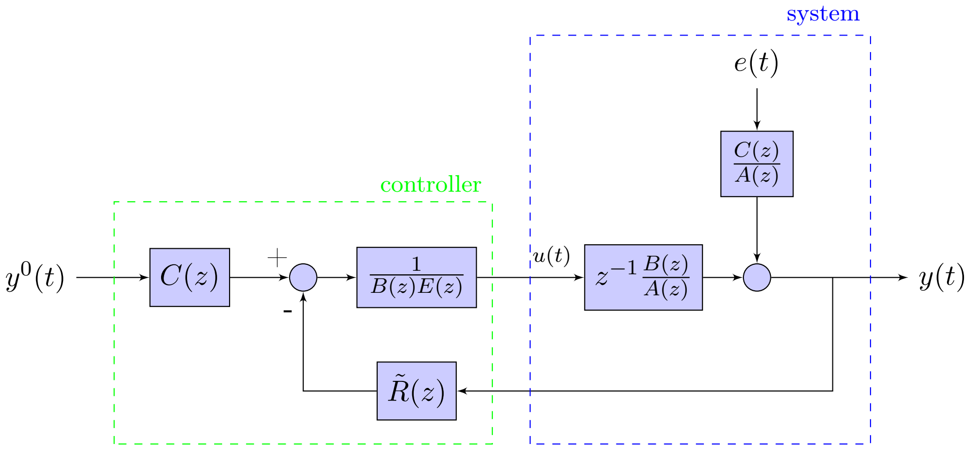
$$\hat{y}(t+k|t) = \frac{B(z)E(z)}{C(z)}u(t) + \frac{\tilde{R}(z)}{C(z)}y(t)$$

Impose $\hat{y}(t+k|t) = y^0(t+k)$, at time $t$ we don't know $y^0(t+k)$, we replace it with $y^0(t)$.

$$\hat{y}(t+k|t) = y^0(t)$$

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

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



### 6.1.1 Stability check

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

$$\chi(z) = z^{-k}B(z)\tilde{R}(z) + B(z)E(z)A(z) = B(z)\left(z^{-k}\tilde{R}(z) + E(z)A(z)\right) = B(z)C(z)$$

The system is asymptotically stable iff:

- All roots of $B(z)$ are stable (minimum phase assumption)
- All roots of $C(z)$ are stable (canonical representation)

### 6.1.2 Performance analysis

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

$$F_{y^0y} = \frac{C(z)\frac{1}{B(z)E(z)}\frac{z^{-k}B(z)}{A(z)}}{1+\frac{1}{B(z)E(z)}\frac{z^{-k}B(z)}{A(z)}\tilde{R}(z)} = z^{-k}$$

$$F_{y^0y} = \frac{\frac{C(z)}{A(z)}}{1+\frac{1}{B(z)E(z)}\frac{z^{-k}B(z)}{A(z)}\tilde{R}(z)} = E(z)$$

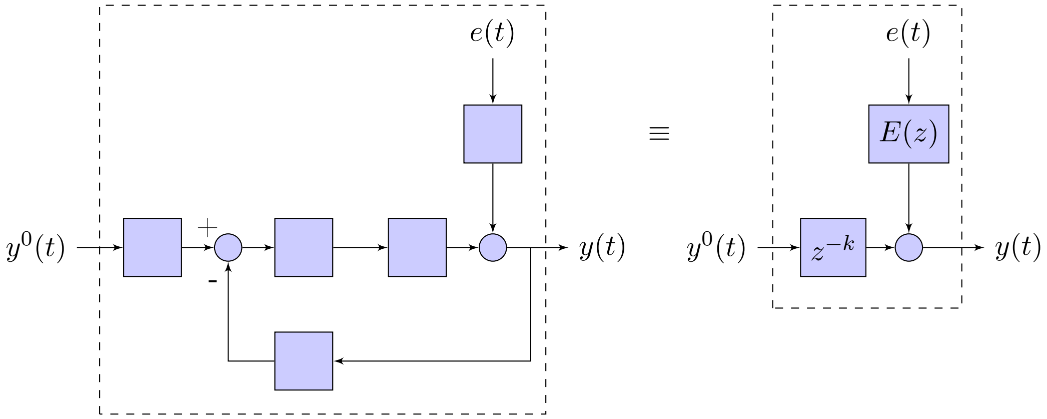
At closed loop the behavior is very simple:

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

$y(t)$ exactly tracks/follows $y^0(t)$ but with $k$ steps of delay and it's disturbed by noise $E(z)e(t)$. This is the best possible solution.

**Remark**

The closed-loop behavior is very simple.



Where are all the poles of the original system?

The M.V. controller *pushes* all the system poles into the non-observable and/or non-controllable parts of the system (it makes internal cancellations). It's not a problem since we verified it's internally asymptotically stable.

## 6.2 Main limits of Minimum Variance Controllers

- 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 $y(t)$

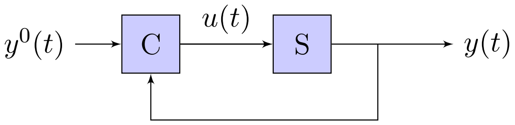
To overcome these limits there is an extension called G.M.V.C. (Generalized Minimum Variance Control). Main difference: extension of the performance index.

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

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

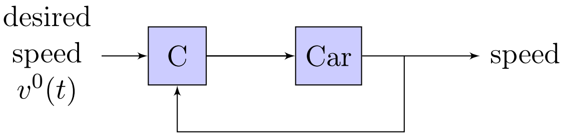
- $P(z)$ is the *reference model*
- $Q(z)$ is a T.F. that makes a penalty to big values of $u(t)$

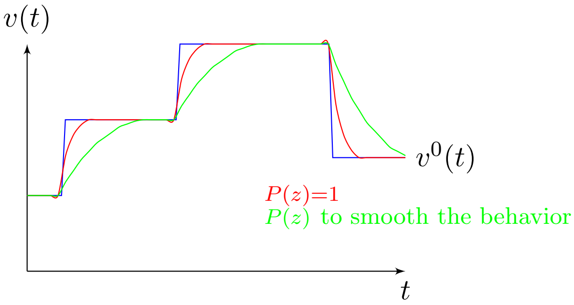
**Remark: Reference model $P(t)$**



The typical goal is to obtain the best possible tracking $y(t) = y^0(t)$, however perfect tracking may not be the best solution, but the best solution is to track a *reference model*: $y(t) = P(z)y^0(t)$.

**Example: Cruise control in a car**




$v(t)$
$v^0(t)$
$P(z)$=1
$P(z)$ to smooth the behavior
$t$

# 7 Recursive Identification

Stefano Dattilo 03/06/2020

**Recall: ARX syatem**

$$y(t) = \frac{B(z)}{A(z)}u(t-1) + \frac{1}{A(z)}e(t) \qquad e(t) \sim WN(m_e, \lambda_e)$$

$$B(z) = b_0 + b_1 z^{-1} + \ldots + b_p z^{-p}$$
$$A(z) = 1 + a_1 z^{-1} + \ldots + a_m z^{-m}$$

$$y(t) = -a_1 y(t-1) - \ldots - a_m y(t-m) + b_0 u(t-1) + \ldots b_p u(t-p-1) + e(t)$$

$$\theta = \begin{bmatrix} a_1 \\ \vdots \\ a_m \\ b_0 \\ \vdots \\ b_p \end{bmatrix} \qquad \phi(t) = \begin{bmatrix} -y(t-1) \\ \vdots \\ -y(t-m) \\ u(t-1) \\ \vdots \\ u(t-p-1) \end{bmatrix} \qquad y(t) = \phi(t)^T\theta + e(t)$$

**Objective** Identify $\theta$ starting from an available dataset.

## 7.1 Least square

$$\hat{\theta}_N = \arg\min_{\theta} \left\{ J_N(\theta) = \frac{1}{N}\sum_{t=1}^{N} (y(t) - \hat{y}(t|t-1,\theta))^2 \right\}$$

We need to find the model predictor $\hat{y}(t|t-1,\theta)$.

$$y(t) = \underbrace{\phi(t)^T\theta}_{\text{known at } t-1} + \underbrace{e(t)}_{\text{unpredictable}}$$

$\hat{y}(t|t-1,\theta) = \phi(t)^T\theta$ is the best possible 1-step predictor.

$$J_N(\theta) = \frac{1}{N}\sum_{t=1}^{N}\big(y(t) - \phi(t)^T\theta\big)^2$$

It is possible to find $\hat{\theta}_N$ (the minimizer) explicitly:

$$\hat{\theta}_N \text{ is such that } \frac{\partial J_N(\theta)}{\partial\theta} = 0$$

$$\hat{\theta}_N = S(N)^{-1}\sum_{t=1}^{N}\phi(t)y(t)$$

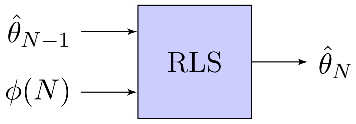
$$S(N) = \sum_{t=1}^{N}\phi(t)\phi(t)^T$$

This procedure has a drawback:

$$\begin{array}{ccc} \{u(1),\dots,u(N)\} & \xrightarrow{\text{new data avail.}} & \{u(1),\dots,u(N),u(N+1)\} \\ \{y(1),\dots,y(N)\} & & \{y(1),\dots,y(N),y(N+1)\} \\ & & \\ \hat{\theta}_N & & \hat{\theta}_{N+1} \end{array}$$

$\hat{\theta}_N$ is not used to compute $\hat{\theta}_{N+1}$, it is necessary to repeat all the computation. The solution is *Recursive Least Square.*

## 7.2 Recursive Least Square



**Advantages**

- Lower computational effort
- You save memory allocation

### 7.2.1 First form

**Objective** $\hat{\theta}_N = f(\hat{\theta}_{N-1}, \phi(N))$

$$\hat{\theta}_N = S(N)^{-1} \sum_{t=1}^{N} \phi(t)y(t) \tag{7.1}$$

$$\sum_{t=1}^{N} \phi(t)y(t) = S(N)\hat{\theta}_N \tag{7.2}$$

$$\hat{\theta}_{N-1} = S(N-1)^{-1} \sum_{t=1}^{N-1} \phi(t)y(t) \tag{7.3}$$

$$\sum_{t=1}^{N-1} \phi(t)y(t) = S(N-1)\hat{\theta}_{N-1} \tag{7.4}$$

$$\sum_{t=1}^{N} \phi(t)y(t) = \sum_{t=1}^{N-1} \phi(t)y(t) + \phi(N)y(N) \tag{7.5}$$

$$\sum_{t=1}^{N} \phi(t)y(t) = S(N-1)\hat{\theta}_{N-1} + \phi(N)y(N) = \text{ equation (7.2)} \tag{7.6}$$

$$S(N)\hat{\theta}_N = S(N-1)\hat{\theta}_{N-1} + \phi(N)y(N) \tag{7.7}$$

We need to find an expression of $S(N-1)$:

$$\begin{aligned}
S(N) = \sum_{t=1}^{N} \phi(t)\phi(t)^T &= \underbrace{\sum_{t=1}^{N-1} \phi(t)\phi(t)^T}_{S(N-1)} + \phi(N)\phi(N)^T \\
S(N-1) &= S(N) - \phi(N)\phi(N)^T \\
S(N)\hat{\theta}_N &= \left(S(N) - \phi(N)\phi(N)^T\right)\hat{\theta}_{N-1} + \phi(N)y(N) \\
S(N)\hat{\theta}_N &= S(N)\hat{\theta}_{N-1} - \phi(N)\phi(N)^T\hat{\theta}_{N-1} + \phi(N)y(N) \\
\hat{\theta}_N &= \hat{\theta}_{N-1} + \underbrace{S(N)^{-1}\phi(N)}_{K(N) \text{ gain}} \underbrace{\left[y(N) - \phi(N)^T\hat{\theta}_{N-1}\right]}_{\epsilon(N) \text{ prediction error}}
\end{aligned}$$

$$\begin{aligned}
\hat{\theta}_N &= \hat{\theta}_{N-1} + K(N)\epsilon(N) \\
K(N) &= S(N)^{-1}\phi(N) \\
\epsilon(N) &= y(N) - \phi(N)^T\hat{\theta}_{N-1} \\
S(N) &= S(N-1) + \phi(N)\phi(N)^T
\end{aligned}$$

This method has a drawback: $S(N) - S(N-1) = \phi(N)\phi(N)^T \geq 0$, so $S(N) \to \infty$. It tends to saturate the numeric precision of the digital computing unit.

### 7.2.2 Second form

**Trick** Normalize w.r.t. $N$

$$\begin{aligned}
S(N) &= S(N-1) + \phi(N)\phi(N)^T \\
R(N) &= \frac{1}{N}S(N) = \frac{1}{N}\frac{N-1}{N-1}S(N-1) + \frac{1}{N}\phi(N)\phi(N)^T \\
&= \frac{N-1}{N}R(N-1) + \frac{1}{N}\phi(N)\phi(N)^T
\end{aligned}$$

$$\begin{aligned}
\hat{\theta}_N &= \hat{\theta}_{N-1} + K(N)\epsilon(N) \\
K(N) &= \frac{1}{N}R(N)^{-1}\phi(N) \\
\epsilon(N) &= y(N) - \phi(N)^T\hat{\theta}_{N-1} \\
R(N) &= \frac{N-1}{N}R(N-1) + \frac{1}{N}\phi(N)\phi(N)^T
\end{aligned}$$

It has a drawback: matrix inversion at each time step. The solution is the recursive least square in third form.

### 7.2.3 Third form

**Lemma 7.1** (Matrix inversion)**.** *Consider 4 matrices $F$, $G$, $H$, $K$ such that:*

- *$F + GHK$ makes sense (dimensionally)*
- *$F$, $H$ and $F + GHK$ square and invertible*

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

In our case $S(N)^{-1} = (\underbrace{S(N-1)}_{F} + \underbrace{\phi(N)}_{G}\underbrace{1}_{H}\underbrace{\phi(N)^T}_{K})^{-1}$

$$S(N)^{-1} = S(N-1)^{-1} - \underbrace{S(N-1)^{-1}}_{\substack{\text{computed at}\\\text{previous step}}}\phi(N)\underbrace{\left[1 + \phi(N)^T S(N-1)\phi(N)\right]^{-1}}_{\text{scalar}}\phi(N)^T S(N-1)^{-1}$$

Define $V(N) = S(N)^{-1}$

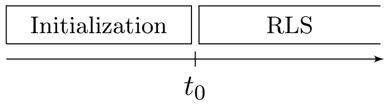
$$V(N) = V(N-1) - \frac{V(N-1)\phi(N)\phi(N)^T V(N-1)}{1 + \phi(N)^T V(N-1)\phi(N)}$$

- $V(N)$ does not tend to zero
- It's no more necessary to invert a matrix

$$\begin{aligned}
\hat{\theta}_N &= \hat{\theta}_{N-1} + K(N)\epsilon(N) \\
K(N) &= V(N)\phi(N) \\
\epsilon(N) &= y(N) - \phi(N)^T\hat{\theta}_{N-1} \\
V(N) &= V(N-1) - \frac{V(N-1)\phi(N)\phi(N)^T V(N-1)}{1+\phi(N)^T V(N-1)\phi(N)}
\end{aligned}$$

**Remark**

RLS is a rigorous version of LS (not an approximation), provided a correct initialization.

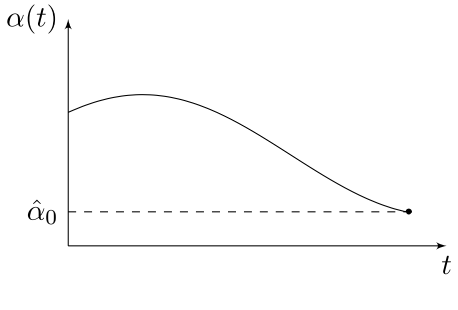


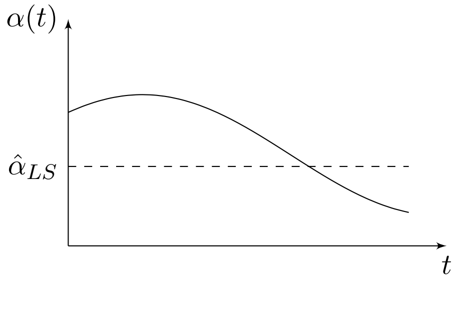
- Collect a first set of data, till $t_0$
- Compute $\hat{\theta}_{t_0}$ and $S(t_0)$ with LS
- Use the recursive to update $\hat{\theta}_t$ and $S(t)$

In practice $\hat{\theta}_0 = 0$ and $S(0) = I$, the error due to the *wrong* initialization will expire with time.

## 7.3 Recursive Least Square with Forgetting Factor

Consider a time varying parameter





$\hat{\alpha}_0$ is the correct estimation at time $N$, but it does not minimizes the objective function $J_N(\alpha) = \frac{1}{N}\sum_{t=1}^{N}\left(y(t) - \hat{y}(t|t-1,\alpha)\right)^2$ because it considers the entire time history of the system.

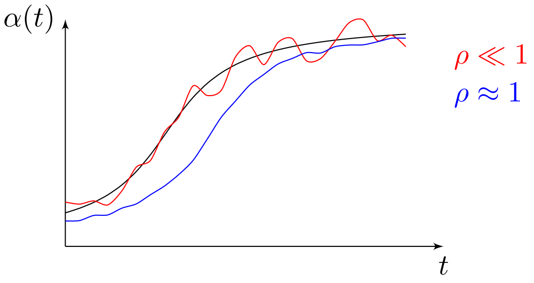
In order to identify a time varying parameter the RLS must be forced to forget old data. The solution is provided by the minimization of $J_N$:

$$J_N(\theta) = \frac{1}{N}\sum_{t=1}^{N}\rho^{N-t}\left(y(t) - \hat{y}(t|t-1,\theta)\right)^2$$

Where $0 \leq \rho \leq 1$ is called *forgetting factor*. If $\rho = 1$ it's the previous formulation. If $\rho < 1$ the old data has less importance than new data.

$$\begin{aligned}\hat{\theta}_N &= \hat{\theta}_{N-1} + K(N)\epsilon(N)\\ K(N) &= S(N)^{-1}\phi(N)\\ \epsilon(N) &= y(N) - \phi(N)^T\hat{\theta}_{N-1}\\ S(N) &= \rho S(N-1) + \phi(N)\phi(N)^T\end{aligned}$$

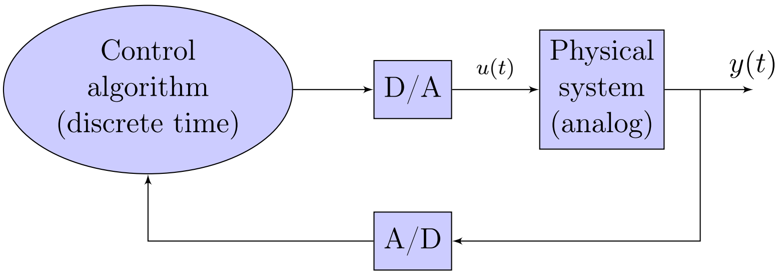
**Remark: Choice of $\rho$**



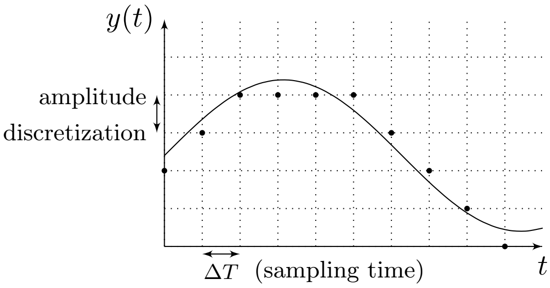
If $\rho \ll 1$ there's high tracking speed but low precision. With $\rho \approx 1$ there's low tracking speed but greater precision.

# 8 Appendix: Discretization of an analog system

Sergio
Savaresi
04/06/2020



### A to D converter



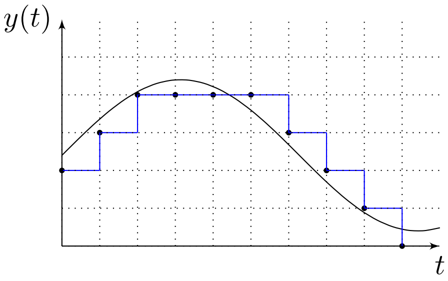
**Time discretization** $\Delta T$ is the sampling time

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

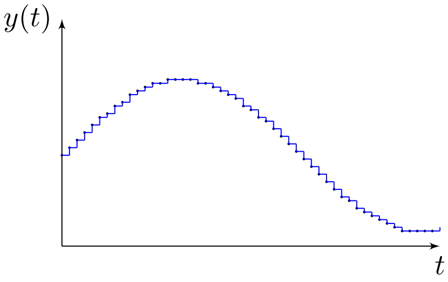
An high quality of A/D converter:

- Can use small $\Delta T$
- High number of levels (16-bits)

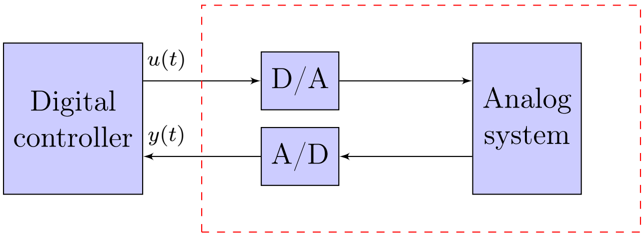
## D to A converter



If $\Delta T$ is sufficiently small, the step-wise analog signal is very similar to a smooth analog signal.



What is the model of the Digital Perspective?



System from a digital perspective

- We make B.B. system identification from measured data: directly estimate a discrete-time model
- We have a physical W.B. model (continuous time), we need to discretize it

The most used approach is the State-Space Transformation.

$$S : \begin{cases} \dot{x} = Ax + bu \\ y = Cx + (Du) \end{cases} \qquad \text{sampling time } \Delta T \qquad S : \begin{cases} x(t+1) = Fx(t) + Gu(t) \\ y(t) = Hu(t) + (Du(t)) \end{cases}$$

Transformation formulas:

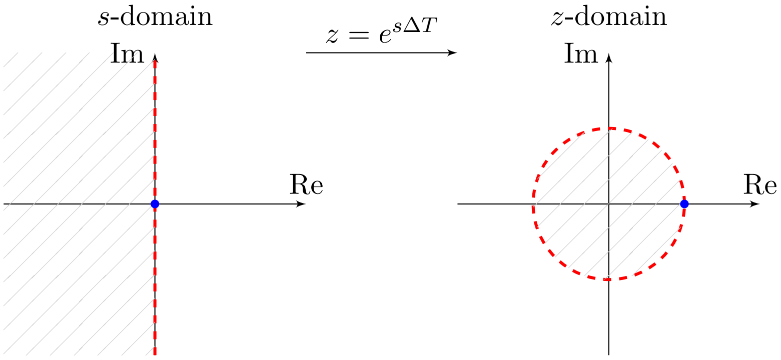
$$
\begin{aligned}
F &= e^{A\Delta T}\\
G &= \int_0^{\Delta T} e^{A\delta} B \, d\delta\\
H &= C\\
D &= D
\end{aligned}
$$

**Remark**

How the poles of the continuous time system are transformed?

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

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



How the zeros in of $S$ in continuous time are transformed into zeros in discrete time?

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

$$G(s) = \frac{\text{polynomial in } s \text{ with } h \text{ zeros}}{\text{polynomial in } s \text{ with } k \text{ poles}} \qquad \text{if } G(s) \text{ is strictly proper: } k > h$$

$$G(z) = \frac{\text{polynomial in } z \text{ with } k-1 \text{ zeros}}{\text{polynomial in } z \text{ with } k \text{ poles}} \qquad G(z) \text{ with relative degree } 1$$

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.

> We need for instance GMVC to design the control system.

Another simple discretization technique frequently used is the discretization of time-derivative $\dot{x}$.

$$\textbf{Eulero backward} \qquad \dot{x} \approx \frac{x(t) - x(t-1)}{\Delta T} = \frac{x(t) - z^{-1}x(t)}{\Delta T} = \frac{z-1}{z\Delta T}x(t)$$

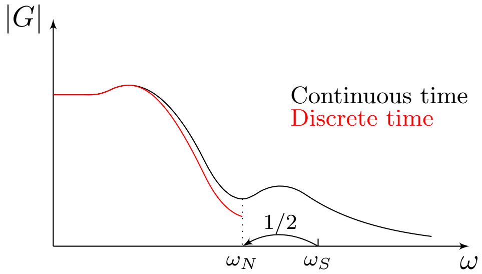
$$\textbf{Eulero forward} \qquad \dot{x} \approx \frac{x(t+1) - x(t)}{\Delta T} = \frac{zx(t) - x(t)}{\Delta T} = \frac{z-1}{\Delta T}x(t)$$

General formula

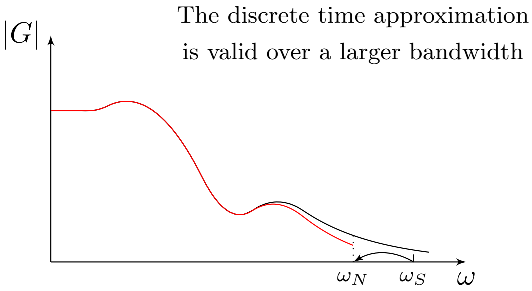
$$\dot{x}(t) = \left[\frac{z-1}{\Delta T}\frac{1}{\alpha z + (1-\alpha)}\right]x(t) \qquad \text{with } 0 \leq \alpha \leq 1$$

- if $\alpha = 0$ it's Eulero Forward
- if $\alpha = 1$ it's Eulero Backward
- if $\alpha = \frac{1}{2}$ it's Tustin method

The critical choice is $\Delta T$ (sampling time). The general intuitive rule is: the smaller $\Delta T$, the better.



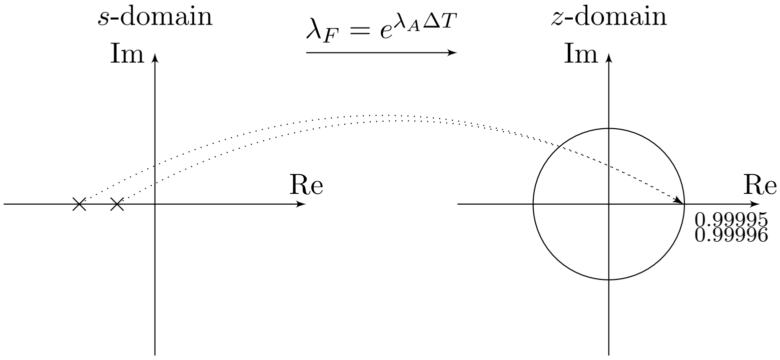
If $\Delta T$ is smaller, $\omega_S$ larger:



$$\Delta T \to f_S = \frac{1}{\Delta T} \qquad \omega_S = \frac{2\pi}{\Delta T} \qquad f_N = \frac{1}{2} f_S \qquad \omega_N = \frac{1}{2}\omega_S$$

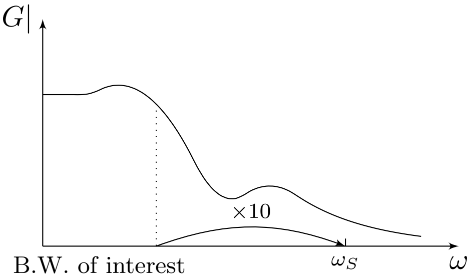
Hidden problems of a too-small $\Delta T$:

- Sampling devices (A/D and D/A) cost
- Computational cost: update an algorithm every $1\mu s$ is much heavier than every $1ms$
- Cost of memory (if data logging is needed)
- Numerical precision *cost* (hidden computational cost)

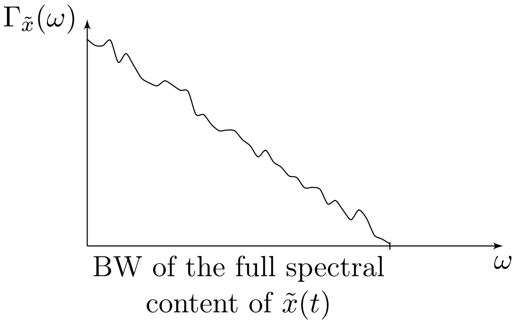


If $\Delta T$ is very small (tends to zero), we squeeze all the poles very closed to $(1, 0)$. We need very high numerical precision (use a lot of digits) to avoid instability.

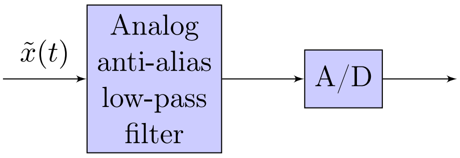
Rule of thumb of control engineers: $f_S$ is between 10 and 20 times the system bandwidth we are interested in.



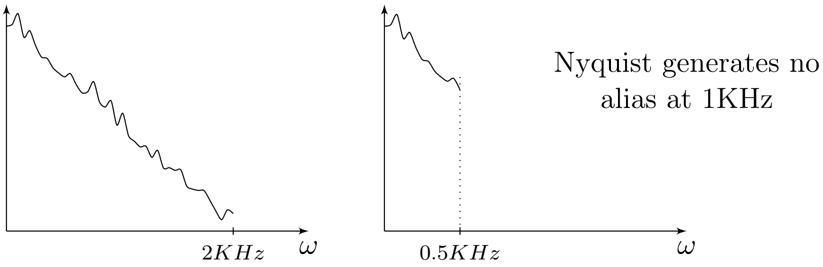
**Remark: Another way of managing the choice of $\Delta T$ w.r.t. the aliasing problem**



The classical way to deal with aliasing is to use anti-alias analog filters

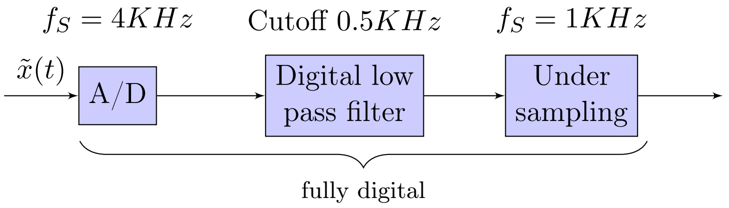


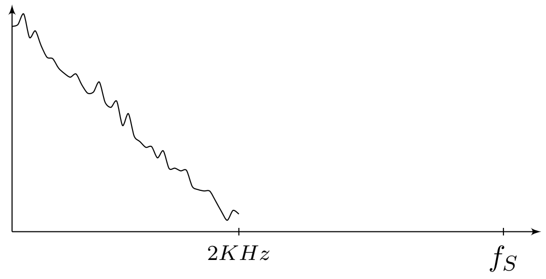
For example, if A/D is at $1KHz$ ($\Delta T = 1ms$)



Nyquist generates no alias at 1KHz

**Full Digital Approach** without analog anti-alias filter


$f_S = 4KHz$
Cutoff $0.5KHz$
$f_S = 1KHz$
$\tilde{x}(t)$
A/D
Digital low pass filter
Under sampling
fully digital


$2KHz$
$f_S$