POLITECNICO DI TORINO

**Master's degree**
**in Computer Engineering**

Master's degree thesis

# Data-Driven Design of Robust Observers for Nonlinear Systems and Application to Satellite Attitude Control


**Supervisors**
prof. Carlo Novara
prof. Mario Milanese

**Candidate**
Marco Russo

Academic Year 2020/2021

# Summary

In control systems, when the state equations are unknown, the design of a state observer usually consists first in identifying the model of the system from experimental data, and then in designing an observer from the identified model. This two-step procedure is generally not optimal. In this thesis, a one-step procedure is presented, where observers are directly designed from experimental data, using neural networks to approximate the nonlinearities in the system. This procedure is here used in two different applications. First, some uncertain linear systems are taken as case studies from research papers where robust linear filters are presented; a nonlinear observer is designed from data and its robustness is compared to those filters. Then, a satellite attitude control system is considered, where the angular velocity cannot be directly measured and is usually estimated using an EKF (extended Kalman filter). If the satellite is used to hook debris, its inertia matrix can vary substantially and in this case robustness is important. A nonlinear observer is then designed from data to estimate the angular velocity and its performance is compared to the EKF.

# Acknowledgements

Thanks to prof. Carlo Novara and to prof. Mario Milanese for their precious suggestions and their immense availability throughout these months, that helped me overcome some technical difficulties and motivated me to work hard and with passion.

# Dedication

A Mimmo, la prima persona per cui abbia mai provato ad essere un buon esempio. Gli anni passano ma resti sempre il mio fratellino, e non sai quante volte sei stato tu un buon esempio per me. Prenditi tutto quello che ti spetta là fuori, lo sai che sei pure meglio di me.

Ai miei genitori, che resteranno sempre i più grandi insegnanti che abbia mai avuto. Grazie per tutti i sacrifici che continuate a fare, non smetterò mai di provare a rendervi fieri di me perché dirvi grazie non basta. Sappiate che vi state laureando anche voi con me.

E a Martina, che ha scommesso su di me sin dal primo giorno e continua a farlo, anche se non dev'essere sempre facile con uno come me. Grazie per essere la mia fan numero uno e per farmi sentire come solo tu riesci. Tu fai la differenza. Rendi tutto mille volte più bello e non avrei voluto nessun'altra persona al mio fianco.

# Contents

**List of Tables** 8

**List of Figures** 9

**I Filter design from data** **11**

**1 Introduction** 13
1.1 Theoretical foundations . . . . . 14
1.1.1 Linear systems . . . . . 16
1.1.2 Nonlinear systems . . . . . 16
1.2 Neural-network-based NARX and NFIR . . . . . 17

**II Design of robust observers with neural networks and linear systems examples** **25**

**2 Design of robust DVSs** 27

**3 Robust DVSs for linear systems** 29
3.1 Comparison with the Shaked filter . . . . . 29
3.2 Comparison with the Duan filter . . . . . 35

**III Spacecraft attitude robust control with neural-network-based observers** **41**

**4 Spacecraft attitude control** 43
4.1 Introduction . . . . . 43
4.2 Rotations . . . . . 44
4.2.1 Euler angles . . . . . 45
4.2.2 Quaternions . . . . . 46

4.3 Attitude kinematics . . . 47
4.4 Attitude dynamics . . . 49
4.5 Attitude control . . . 51
4.5.1 Sensors . . . 51
4.5.2 Actuators . . . 52
4.5.3 Control approaches . . . 52
4.6 State equations . . . 53

**5 Design of a DVS for attitude control** 55
5.1 Case specifications . . . 55
5.2 Training phase . . . 56
5.3 Closed-loop and open-loop performance . . . 60

**6 Conclusions** 65

**A Linear Kalman Filter** 67

**B Extended Kalman Filter** 69

# List of Tables


3.1 Performances varying only $\delta$ (Shaked case) . . . . . 32
3.2 Performances varying only $\sigma_w^2$ and $\sigma_\nu^2$ (Shaked case) . . . . . 33
3.3 Performances with uniformly distributed noises (Shaked case) . . . 34
3.4 Performances varying only $\delta$ (Duan case) . . . . . 38
5.1 Average error variance of DVS and EKF in closed-loop . . . . . 60
5.2 Average error variance of DVS and EKF in open-loop . . . . . 61

# List of Figures


1.1 NARX filter structure . . . 18
1.2 Sigmoid neural network . . . 19
1.3 Wavelet neural network . . . 20
1.4 Open-loop neural network . . . 21
1.5 Closed-loop neural network . . . 21
3.1 Simulink scheme of the system (Shaked case) . . . 30
3.2 Plot of the nonlinearity of the DVS (Shaked case) . . . 31
3.3 Validation accuracy with $\delta = 0$ (Shaked case) . . . 32
3.4 Validation accuracy with $\delta = -0.3, +0.3$ (Shaked case) . . . 33
3.5 Validation accuracy with $\delta = -0.5, +0.5$ (Shaked case) . . . 34
3.6 Estimation error with $x_0 = [1000; 1000]$ (Shaked case) . . . 35
3.7 Simulink scheme of the system (Duan) . . . 37
3.8 Plot of the nonlinearity of the DVS (Duan case) . . . 37
3.9 Validation accuracy with $\delta = 0$ (Duan case) . . . 38
3.10 Validation accuracy with $\delta = -0.45, +0.45$ (Duan case) . . . 39
3.11 Validation accuracy with $\delta = -0.5, +0.5$ (Duan case) . . . 40
4.1 A rendering of ELSA-d (Author: Astroscale, CC-BY-SA) . . . 44
4.2 Representing a point with respect to two different reference frames . 45
4.3 Representing a body point with respect to two different reference frames . . . 49
4.4 Sodern's HYDRA-M star tracker (Author: Christian Lafont, CC-BY-SA) . . . 52
5.1 Angular speed behaviour in open-loop with uniform random input . 56
5.2 Open loop Simulink scheme used for training . . . 57
5.3 Time Delay Neural Network used for the DVS . . . 57
5.4 Training and validation error during training for the first DVS . . . 59
5.5 Plot of the nonlinearity of the DVS (Spacecraft) . . . 59
5.6 Deep Time Delay Neural Network used for the DVS . . . 60
5.7 Spacecraft attitude in closed-loop with $k = 3$ using a T-DVS (top) and an EKF (bottom) . . . 62

5.8 Estimation error in closed-loop with $k = 3$ using a T-DVS (top) and an EKF (bottom) . . . . . 63

# Part I

# Filter design from data

# Chapter 1

# Introduction

The design of filters and state observers in control systems is a frequent task when some unmeasurable or noise-corrupted variables are needed as inputs to the controller. In most cases, however, the equations of the system to be filtered are unknown and the common paradigm consists in first identifying the system from experimental data, and then designing a filter on the identified system. This is a two-step procedure that is characterized by an evident problem: since the identified model is just an approximation of the system, the resulting filter is only optimal for this approximation, not for the actual system, on which its performance may decrease substantially. Furthermore, when the system is nonlinear, designing an optimal filter is a very difficult task ([1]) and only approximate filter can be derived most of the times ([2]). There is also the case of uncertain systems, where the design of robust filters requires the knowledge of the uncertainty model (i.e. a model of the parametric uncertainties), but when the system is nonlinear it is very hard to obtain such a model. Even when the system is linear, the linear Kalman filter designed for the nominal system is not robust and it has large variance when the system is perturbed; in this case, other robust filter design methods are proposed in the literature ([3], [4]) but their robustness is still not high.
In [5] a one-step procedure is proposed to overcome the drawbacks of the two-step approach. Here the filter is obtained directly from experimental data, without first identifying the system. The filter obtained with this procedure is called a Direct Virtual Sensor (DVS). The advantage of this approach is that the performance is usually better than the two-step one, especially in the case of uncertain systems.
In this thesis, this procedure is analyzed in the parametric-statistic approach (stochastic noises and parametric filter structure) and it is implemented to design robust observers by means of Neural Networks and, after a theoretical introduction of its foundations, it is first shown that, even for linear uncertain systems, the most robust filter is nonlinear, comparing the performances with the results in [3] and [4]; then, a nonlinear uncertain system is considered (a spacecraft attitude

control system) and a DVS is used to estimate some unmeasurable states and its estimation accuracy is compared to the one of an extended Kalman filter. Also in this case it is shown that the DVS has the best accuracy and robustness. The work in this thesis therefore confirms the superiority of the one-step procedure to the two-step one for both linear and nonlinear systems and proposes a design method that guarantees better performances and robustness than what can be found on the literature for filter design, and the consequences are of paramount importance for situations where high accuracy is needed for precision tasks in uncertain scenarios and where inaccuracies can lead to high costs and even mission failures.

## 1.1 Theoretical foundations

Following the notation in [5], let S be a discrete-time system, described by the equations

$$\mathbf{x}^{t+1} = \mathbf{F}(\mathbf{x}^t, \tilde{\mathbf{u}}^t) + \mathbf{w}_x^t \tag{1.1}$$
$$\tilde{\mathbf{y}}^t = \mathbf{H}_y(\mathbf{x}^t, \tilde{\mathbf{u}}^t) + \mathbf{w}_\mathbf{y}^t \tag{1.2}$$
$$\tilde{\mathbf{z}}^t = \mathbf{H}_z(\mathbf{x}^t, \tilde{\mathbf{u}}^t) + \mathbf{w}_z^t, \tag{1.3}$$

where

- $\mathbf{x}^t \in \mathbb{R}^{n_x}$ is the state
- $\tilde{\mathbf{u}}^t \in \mathbb{R}^{n_u}$ is the known input
- $\tilde{\mathbf{y}}^t \in \mathbb{R}^{n_y}$ is a measured output
- $\tilde{\mathbf{z}}^t \in \mathbb{R}^{n_z}$ is a quantity to estimate
- $\mathbf{w}_x^t$ is the process noise and $\mathbf{w}_\mathbf{y}^t, \mathbf{w}_z^t$ are output noises.

The objective is designing a filter that, using the inputs $(\tilde{\mathbf{u}}^\tau, \tilde{\mathbf{y}}^\tau)$, $\tau \leq t$, gives an estimate $\hat{\mathbf{z}}^t$ as output. Such a filter is indicated in the paper as **Direct Virtual Sensor (DVS)**. In a **parametric-stochastic approach**, the noises $\mathbf{w}_x^t$ and $\mathbf{w}_\mathbf{y}^t$ are assumed to be stochastic, a parametric filter structure is considered and the objective is minimizing $Var(\tilde{\mathbf{z}}^t - \hat{\mathbf{z}}^t)$ for any $t$.
Let us now suppose we have a set of collected data $\{\tilde{\mathbf{u}}^t, \tilde{\mathbf{y}}^t, \tilde{\mathbf{z}}^t, t = 1,2,\dots,T\}$, the functions $\boldsymbol{f}, H_y, H_z$ are unknown and $(\mathbf{F}, \mathbf{H}_y)$ is observable in the sense of [5]. In the following paragraphs, the two-step approach and the one-step approach are compared.
As previously stated, the two-step approach consists in first identifying the system

and then designing a filter on the identified system. In the first step, we select a *parametric model structure* that defines the following model set:

$$\mathcal{M} := \{M(\boldsymbol{\theta}_M) : \boldsymbol{\theta}_M \in \boldsymbol{\theta}_M\}, \tag{1.4}$$

where $\boldsymbol{\theta}_M$ is a compact subset of $\mathbb{R}^{n_{\boldsymbol{\theta}M}}$ and $n_{\boldsymbol{\theta}M}$ is the number of parameters of the model structure. The model structure selection corresponds to the choice of the functions that approximate the system behavior; for instance, it could be a neural network with a defined structure, a sum of polynomials up to a certain order, etc. Using the training set

$$D_M := \{\tilde{\mathbf{u}}^t, (\tilde{\mathbf{y}}^t, \tilde{\mathbf{z}}^t), t = 1, \dots, T\}, \tag{1.5}$$

a model $\hat{M} = M(\hat{\boldsymbol{\theta}}_M)$ of S is identified from $D_M$, where $\tilde{\mathbf{u}}^t$ is considered as the input of $\hat{M}$ and $(\tilde{\mathbf{y}}^t, \tilde{\mathbf{z}}^t)$ as the outputs. The model is identified selecting the $\hat{\boldsymbol{\theta}}_M$ such that

$$\hat{\boldsymbol{\theta}}_M = \arg \min_{\boldsymbol{\theta}_M \in \boldsymbol{\theta}_M} \frac{1}{T} \sum_{i=1}^{T} \frac{1}{2} \|(\tilde{\mathbf{y}}^t, \tilde{\mathbf{z}}^t) - (\mathbf{y}_M^t, \hat{\mathbf{z}}_M^t)\|_2^2, \tag{1.6}$$

where $(M^t, \hat{\mathbf{z}}_M^t)$ is the prediction given by $M(\boldsymbol{\theta}_M)$. In the second step, then, a minimum-variance filter $\hat{K} = K(\hat{\boldsymbol{\theta}}_M)$ is designed for $\hat{M}$, in order to give an output estimate $\hat{\mathbf{z}}^t$ of $\tilde{\mathbf{z}}^t$ using $(\tilde{\mathbf{u}}^\tau, \tilde{\mathbf{y}}^\tau), \tau \leq t$.
In the one-step approach proposed in the paper, instead, we select a parametric filter structure that defines the following set:

$$\mathcal{V} := \{V(\boldsymbol{\theta}_V) : \boldsymbol{\theta}_V \in \boldsymbol{\theta}_V\}, \tag{1.7}$$

where $\boldsymbol{\theta}_V$ is a compact subset of $\mathbb{R}^{n_{\boldsymbol{\theta}V}}$ and $n_{\boldsymbol{\theta}V}$ is the number of parameters. Using the training set

$$D_V := \{(\tilde{\mathbf{u}}^t, \tilde{\mathbf{y}}^t), \tilde{\mathbf{z}}^t, t = 1, \dots, T\}, \tag{1.8}$$

a filter $\hat{V} = V(\hat{\boldsymbol{\theta}}_V)$ for S is directly identified from $D_V$, where $(\tilde{\mathbf{u}}^t, \tilde{\mathbf{y}}^t)$ are considered as the inputs of $\hat{V}$ and $\tilde{\mathbf{z}}^t$ as the output (the difference between $D_V$ and $D_M$ is evident). The filter is identified selecting the $\hat{\boldsymbol{\theta}}_V$ such that

$$\hat{\boldsymbol{\theta}}_V = \arg \min_{\boldsymbol{\theta}_V \in \boldsymbol{\theta}_V} \frac{1}{T} \sum_{t=1}^{T} \frac{1}{2} \|\tilde{\mathbf{z}}^t - \hat{\mathbf{z}}_V^t\|^2, \tag{1.9}$$

where $(\tilde{\mathbf{u}}^t, \tilde{\mathbf{y}}^t)$ is the input of the filter and $\hat{\mathbf{z}}_V^t$ is the output of the filter.
Note that, while the structure of $\hat{K}$ can not be chosen as it depends on the identified model structure, the structure of $\hat{V}$ can instead be chosen, without even identifying a model of S. This is where the name *Direct Virtual Sensor* comes from for $\hat{V}$, whereas $\hat{K}$ is a *Model-based Virtual Sensor* instead. Let us now discuss the properties of DVSs both for linear and nonlinear systems.

### 1.1.1 Linear systems

When the system S is linear, let us assume that a linear filter structure $V(\boldsymbol{\theta}_V)$ is selected and, at the same time, a linear Kalman-filter $\hat{K}$ is designed to estimate $\hat{\mathbf{z}}^t$ on the basis of an identified model $\hat{M}$ of S belonging to a linear model structure $M(\boldsymbol{\theta}_M)$. If the model structure $M(\boldsymbol{\theta}_M)$ is of order $n_M$, so is $K(\boldsymbol{\theta}_M)$; therefore, if the selected structure $\mathcal{V}(\boldsymbol{\theta})$ is of order $n_M$, we have $K(\boldsymbol{\theta}_M) \in \mathcal{V}$.
When the assumptions in [5], Section 3.1 are verified, the following theorem holds:

*Theorem 1*
With probability 1 as $T \to \infty$,

1. $\hat{V} = \arg\min_{V(\boldsymbol{\theta}_V)} \mathbb{E}(\|\tilde{\mathbf{z}}^t - \hat{\mathbf{z}}_V^t\|_2^2)$ ($\hat{V}$ is the minimum-variance filter among all filters belonging to $\mathcal{V}$)
2. If $\hat{K} \in \mathcal{V}$, then $\mathbb{E}(\|\tilde{\mathbf{z}}^t - \hat{\tilde{\mathbf{z}}}_V^t\|_2^2) \leq \mathbb{E}(\|\tilde{\mathbf{z}}^t - \hat{\tilde{\mathbf{z}}}_K^t\|_2^2)$
3. If $S = M(\boldsymbol{\theta}_M^0) \in \mathcal{M}$ and $K(\boldsymbol{\theta}_M^0) \in \mathcal{V}$, then $\hat{V}$ is a minimum variance filter among all linear causal filters mapping $(\tilde{\mathbf{u}}^\tau, \tilde{\mathbf{y}}^\tau) \to \tilde{\mathbf{z}}^\tau, \tau \leq t$
4. If $S = M(\boldsymbol{\theta}_M^0) \in \mathcal{M}$, $K(\boldsymbol{\theta}_M^0) \in \mathcal{V}$, $M(\boldsymbol{\theta}_M)$ is globally identifiable, S is stable, and the data are informative enough, then $\mathbb{E}(\|\tilde{\mathbf{z}}^t - \hat{\tilde{\mathbf{z}}}_V^t\|_2^2) = \mathbb{E}(\|\tilde{\mathbf{z}}^t - \hat{\tilde{\mathbf{z}}}_K^t\|_2^2)$

The previous theorem states in particular that, under the hypothesis of the theorem, the filter $\hat{V}$ is always better than $\hat{K}$, even when S is unstable; the only case where $\hat{K}$ has the same estimation error variance of $\hat{V}$ is when the assumptions in point 4 are satisfied (in particular, the stability of S and the fact that $S \in M$, which means that the model structure perfectly models S).

### 1.1.2 Nonlinear systems

When the system S is nonlinear, let us assume that a nonlinear filter structure $V(\boldsymbol{\theta}_V)$ is selected satisfying condition M1 (see Appendix of [5]) and is related to the parametric nonlinear regression form

$$\hat{\mathbf{z}}_V^t = \boldsymbol{f}_V(\boldsymbol{\theta}_V, \hat{\mathbf{z}}_V^{t-1}, \ldots, \hat{\mathbf{z}}_V^{t-n_V}, \tilde{\mathbf{y}}^t, \ldots, \tilde{\mathbf{y}}^{t-n_V}, \tilde{\mathbf{u}}^t, \ldots, \tilde{\mathbf{u}}^{t-n_V}). \tag{1.10}$$

At the same time, a nonlinear minimum variance filter $\hat{K}$ is designed to estimate $\hat{\mathbf{z}}^t$ on the basis of an identified model $\hat{M}$ belonging to a nonlinear model structure $M(\boldsymbol{\theta}_M)$ that satisfies M1. When the assumptions in [5], Section 3.2 are verified, the following theorem holds:

*Theorem 2*
With probability 1 as $T \to \infty$,

1. $\hat{V} = \arg\min_{V(\boldsymbol{\theta}_V)} \mathbb{E}(\|\tilde{\mathbf{z}}^t - \hat{\mathbf{z}}_V^t\|_2^2)$ ($\hat{V}$ is the minimum-variance filter among all filters belonging to $\mathcal{V}$)
2. If $\hat{K} \in \mathcal{V}$, then $\mathbb{E}(\|\tilde{\mathbf{z}}^t - \hat{\tilde{\mathbf{z}}}_V^t\|_2^2) \leq \mathbb{E}(\|\tilde{\mathbf{z}}^t - \hat{\tilde{\mathbf{z}}}_K^t\|_2^2)$
3. If $S = M(\boldsymbol{\theta}_M^0) \in \mathcal{M}$ and $K(\boldsymbol{\theta}_M^0) \in \mathcal{V}$, then $\hat{V}$ is a minimum variance filter.

Note that, differently from the linear case, the minimum variance filter $\hat{K}$ cannot generally be computed, but only approximated; therefore, the advantages of the one-step method over the two-step one are even more significant.

To summarize, $\boldsymbol{f}_V$ can be a linear function in the case of linear systems, but, more in general, it is a nonlinear function chosen according to the standard parametrizations found in the literature, that vary from radial basis functions to **neural networks** ([6]). It is important, therefore, to notice that this method is generalizable to any form of nonlinear approximation for $\boldsymbol{f}$, and that neural networks are just one of these forms. A DVS, in other words, can use neural networks to approximate the nonlinearity of the system, but also polynomial functions etc., and the DVS method is the same for any choice. The following section explores in depth the structure of the filter when the neural-network-based NARX model is chosen.

## 1.2 Neural-network-based NARX and NFIR

The design of a filter following the procedure that was explained above considers the variable to estimate as a time series to be forecast using other other input time seris given as inputs. There are many ways to model a time series, one of which is the **nonlinear autoregressive exogenous (NARX) model**. According to this model, the filter takes an input $u$ and an output $\mathbf{y}$ and estimates

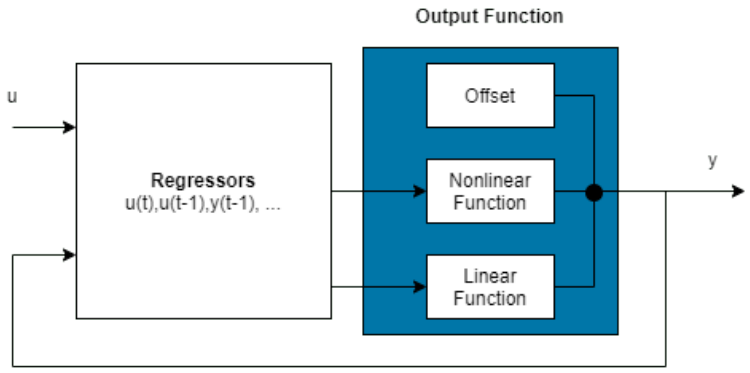
$$\hat{\mathbf{y}}_V^t = \boldsymbol{f}(\mathbf{y}^{t-1}, \mathbf{y}^{t-2}, \dots, \mathbf{y}^{t-n_a}, \mathbf{u}^t, \mathbf{u}^{t-1}, \dots, \mathbf{u}^{t-(n_b+1)}). \tag{1.11}$$

The quantities $\mathbf{y}^{t-1}, \mathbf{y}^{t-2}, \dots, \mathbf{y}^{t-n_a}, \mathbf{u}^t, \mathbf{u}^{t-1}, \dots, \mathbf{u}^{t-(n_b+1)}$ are the model regressors and the function $\boldsymbol{f}$ is the output function. The term "autoregressive" comes from the presence of the past estimated values of the same series among the regressors, while the term "exogenous" come from the presence of an exogenous input series $u$ among the regressors. The notation is not to be confused with the one used above; indeed, for a system in the form described above, we would have $\mathbf{u}^t = [\tilde{\mathbf{u}}^t, \tilde{\mathbf{y}}^t]$ and $\mathbf{y}^t = \tilde{\mathbf{z}}^t$. Note that in the previous equations the output estimate at time $t$ is done using the *measured* outputs at previous time steps; this is called a *one-step prediction*. If, instead, the past *estimated* values are used, i.e.

$$\hat{\mathbf{y}}_V^t = \boldsymbol{f}(\hat{\mathbf{y}}^{t-1}, \hat{\mathbf{y}}^{t-2}, \dots, \hat{\mathbf{y}}^{t-n_a}, \mathbf{u}^t, \mathbf{u}^{t-1}, \dots, \mathbf{u}^{t-(n_b+1)}), \tag{1.12}$$

then we have a *multi-step prediction*. Assuming that the real $\mathbf{y}$ is known at training time, the training is different depending on the chosen type of prediction. If the filter performs a one-step prediction, then the training shall be focused on prediction, i.e. the real $\mathbf{y}$ has to be used as regressor. If, instead, the filter performs a multi-step prediction, the training shall be focused on *simulation*, and the estimated $\hat{\mathbf{y}}$ is used as regressor, whereas the real $\mathbf{y}$ is only used to compare the filter output $\hat{\mathbf{y}}(t)$ and the real output $\mathbf{y}(t)$. Note that, while in the training phase we assume to be able to measure $\mathbf{y}$ to use it for training (for example by simulating the system), at runtime instead it may be unmeasurable; in that case, multi-step prediction is the only way. As mentioned before, the structure of $\boldsymbol{f}$ can be any nonlinear expression, for instance polynomials. The case where $\boldsymbol{f}$ is the output function of a neural network is considered in the following. Using the specifications of the System Identification Toolbox in Matlab (used for this work, together with the Deep Learning Toolbox) the output function follows the structure in Figure 1.1 and it is basically a neural network whose output function is

Figure 1.1. NARX filter structure




$$f(\mathbf{x}) = \mathbf{L}^T(\mathbf{x} - \mathbf{r}) + g(\mathbf{Q}(\mathbf{x} - \mathbf{r})) + d, \tag{1.13}$$

where $\mathbf{x}$ is the vector of regressors, $\mathbf{L}^T(\mathbf{x} - \mathbf{r})$ is the output of the linear function block, $g(\mathbf{Q}(\mathbf{x} - \mathbf{r}))$ is the output of the nonlinear function block, $\mathbf{r}$ is the vector of the mean of the regressors, $\mathbf{Q}$ is a projection matrix that makes the calculations well conditioned and $d$ is a scalar offset added to the output. $g(\mathbf{x})$ is usually a sum of $n$ nonlinear units:

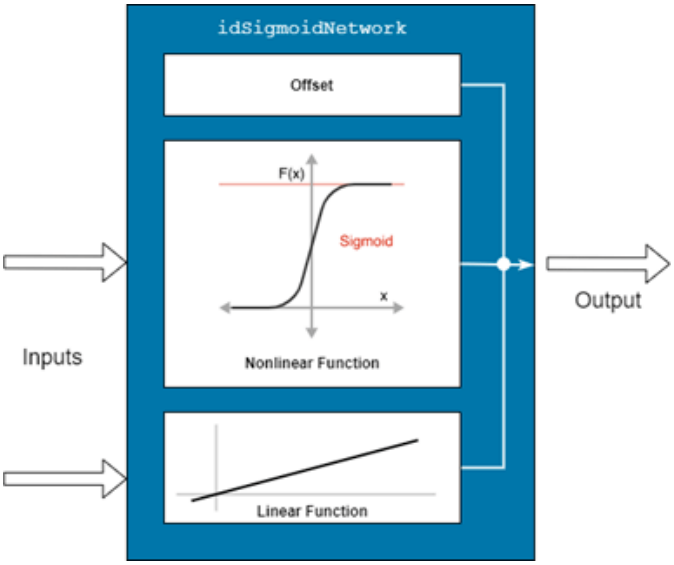
$$\mathbf{g}(\mathbf{x}) = \sum_{k=1}^{n} \alpha_k \kappa(\boldsymbol{\beta}_k^T(\mathbf{x} - \boldsymbol{\gamma}_k)), \tag{1.14}$$

where $\boldsymbol{\beta}_k$ is a vector (so $\boldsymbol{\beta}_k^T(\mathbf{x} - \boldsymbol{\gamma}_k)$ is a scalar) and $\kappa(s)$ is generally chosen to be the **sigmoid function**, i.e.

$$\kappa(s) = \frac{1}{1 + e^{-s}}. \tag{1.15}$$

The output function is then represented by the neural network in Figure 1.2. More

Figure 1.2. Sigmoid neural network




specifically, this neural network maps the input $\mathbf{x}(t) = [\mathbf{x}_1(t), \mathbf{x}_2(t), \ldots, \mathbf{x}_m(t)]^T$ to the scalar

$$y(t) = y_0 + (\mathbf{x}(t) - \overline{\mathbf{x}})^T \mathbf{PL} + S(\mathbf{x}(t)), \tag{1.16}$$

where

- $\mathbf{x}(t) \in \mathbb{R}^m$ is the vector of regressors with mean $\overline{x}$
- $y_0 \in \mathbb{R}$ is a scalar offset
- $P \in \mathbb{R}^{m,p}$, $m \geq p$, is a projection matrix ($p$ is the number of linear weights)
- $L \in \mathbb{R}^p$ is a vector of weights
- $S(\mathbf{x})$ is a sum of dilated and translated sigmoid functions:

$$S(\mathbf{x}) = \sum_{i=1}^{n} s_i f((\mathbf{x} - \overline{\mathbf{x}})^T \mathbf{Q}\mathbf{b}_i + c_i), \tag{1.17}$$

  where

  - $n$ is the number of units,
  - $\mathbf{Q} \in \mathbb{R}^{m,q}$, $m \geq q$ is a projection matrix,
  - the $s_i \in \mathbb{R}$ are the output coefficients,

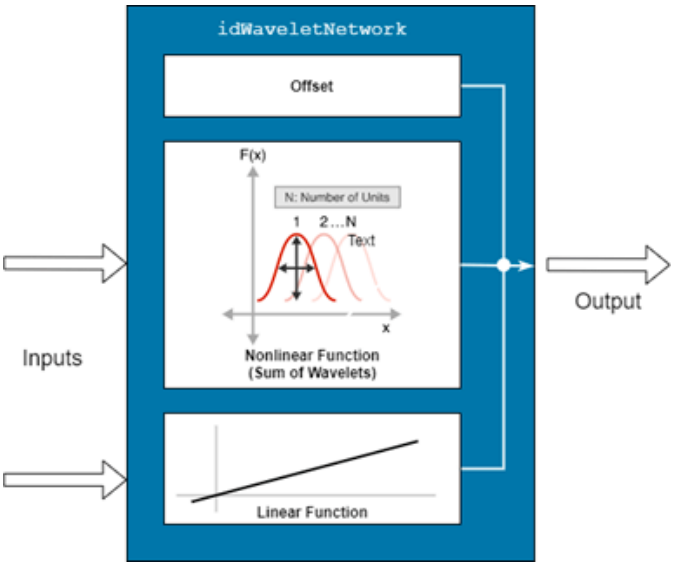
– the $\mathbf{b}_i \in \mathbb{R}^q$ are the dilation coefficients,

– the $c_i \in \mathbb{R}$ are the translations,

– $f(z)$ is the sigmoid functions as defined above.

Other neural network structures that can be considered are the **wavelet networks**, where the nonlinear function operates on radial combinations of inputs. The structure can be seen in Figure 1.3.

Figure 1.3. Wavelet neural network




Here, the input $\mathbf{x}(t) = [\mathbf{x}_1(t), \mathbf{x}_2(t), \ldots, \mathbf{x}_m(t)]^T$ is mapped to the scalar

$$y(t) = y_0 + (\mathbf{x}(t) - \overline{\mathbf{x}}^T \mathbf{P}\mathbf{L} + W(\mathbf{x}(t)) + S(\mathbf{x}(t)), \tag{1.18}$$

where

- $\mathbf{x}(t), \overline{\mathbf{x}}, y_0, \mathbf{P}, \mathbf{L}$ are defined as above,
- $W(\mathbf{x})$ is a sum of $d_w$ dilated and translated wavelets as in the following equation:

$$W(\mathbf{x}) = \sum_{i=1}^{d_w} w_i f_w(\mathbf{b}_i(\mathbf{x} - \overline{\mathbf{x}})^T \mathbf{Q} - \mathbf{c}_i), \tag{1.19}$$

where $f_w(\mathbf{x}) = e^{\frac{-\mathbf{x}^T\mathbf{x}}{2}}$,

- $S(\mathbf{x})$ is a sum of $d_s$ dilated and translated scaling functions (scalelets) as in the following equation

$$S(\mathbf{x}) = \sum_{i=1}^{d_s} s_i f_s(\mathbf{b}_i(\mathbf{x} - \overline{\mathbf{x}})^T \mathbf{Q} - \mathbf{e}_i), \tag{1.20}$$

  where $f_s(\mathbf{x}) = (dim(\mathbf{x}) - \mathbf{x}\mathbf{x}^T)e^{\frac{-\mathbf{x}^T\mathbf{x}}{2}}$

As previously stated, the difference between one-step prediction and multi-step prediction lies in the use of either $\mathbf{y}$ or $\hat{\mathbf{y}}$ as a regressor. The network structure changes accordingly; for instance, using the *narxnets* in the Deep Learning Toolbox, if we choose the one-step prediction then the network must be in open-loop mode, as in Figure 1.4; instead, if we choose the multi-step prediction then the network must be in *closed-loop* mode, as in Figure 1.5. The difference is evident and it lies in the feedback of the estimated $\hat{\mathbf{y}}$ in the second case, whereas in the first case only the real measurements $\mathbf{y}$ are used.

Figure 1.4. Open-loop neural network

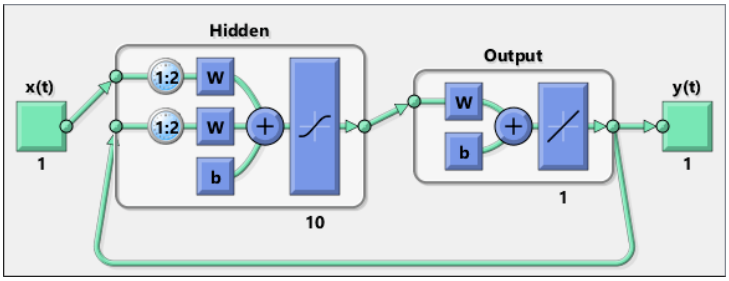
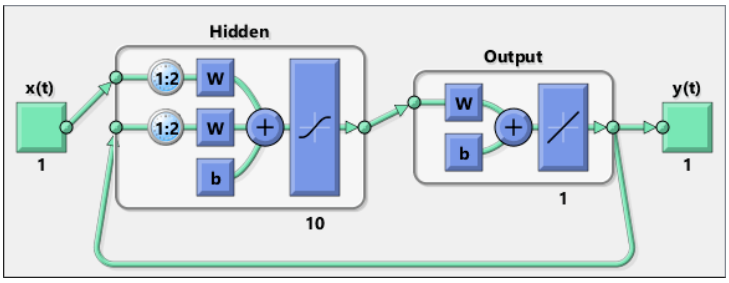



Figure 1.5. Closed-loop neural network




As to the nonlinear function, there are other functions that can be used, but in this thesis the **sigmoid function** was used. Furthermore, to ensure stability of the observer, the autoregressive nature of NARX was discarded, by not putting

any $\hat{\mathbf{y}}(t-\tau), \tau = 1, \dots$ among the regressors: the input of the filter is then only made of a number of current and delayed values of the input $u$, so that, in general, the filter V estimates $\mathbf{y}$ according to this equation:

$$\hat{\mathbf{y}}_V^t = \boldsymbol{f}(\boldsymbol{\theta}; \mathbf{u}^t, \mathbf{u}^{t-1}, \dots, \mathbf{u}^{t-(n_b+1)}), \tag{1.21}$$

where $\boldsymbol{f}(\mathbf{x})$ is the output of the sigmoid network and $\boldsymbol{\theta}$ is the vector of the parameters of the neural network, calculated during the training phase using, for instance, the **nonlinear least squares** as a performance metrics. Note that by removing any $\hat{\mathbf{y}}(t-\tau)$ among the regressors, the filter now becomes a **nonlinear finite impulse response (NFIR) filter**. The full NARX model is instead a *nonlinear infinite impulse response (NIIR)* model. It is reminded that the difference between a FIR and an IIR is that, when the input is the Dirac's delta function, the output response of the FIR goes to 0 after a finite time, whereas the one of the IIR does not. In other words, an NFIR can be obtained by using a NARX and removing the $\mathbf{y}$ estimates among the regressors, and this is the design choice of this thesis. Given the lack of autoregression in the chosen filter structure, the results of *Theorem 1* and *Theorem 2* are not guaranteed; however, it is shown in the following sections that even without autoregression the DVS still outperforms the Kalman filter when the system S is linear and a parametric perturbation is present (the Kalman filter is better only on the nominal system); furthermore, when the system S is nonlinear, the non-autoregressive DVS outperforms the extended Kalman filter even on the nominal system. This is an important result, since the linear Kalman filter and the EKF are autoregressive. As a final remark, let $\boldsymbol{f}^0$ be the true function that we want to approximate with $\boldsymbol{f}(\boldsymbol{\theta}; \mathbf{x}) = \sum_{i=1}^{r} \alpha_i \sigma_i(\mathbf{x})$, where $\mathbf{x}$ is the vector of the regressors and the various $\sigma_i(\mathbf{x})$ are the *basis*. The basis can be fixed (for instance polynomials in $\mathbf{x}$) or tunable (for instance, using neural networks, the sigmoid functions in $\mathbf{x}$). It is clear that the objective is to find the $\sigma_i$ and $\boldsymbol{\theta}$ such that $\boldsymbol{f}(\boldsymbol{\theta}; \mathbf{x}) \to \boldsymbol{f}^0(\mathbf{x})$ as $r \to \infty$. It is proven that the number $r$ of basis needed to accurately approximate $\boldsymbol{f}^0$ with $\boldsymbol{f}$ grows *exponentially* with the number of regressors $n$ if the basis is fixed (this problem is known as the **curse of dimensionality**). If the basis is tunable, however, and $\boldsymbol{f}^0$ satisfies some regularity conditions, $r$ grows *linearly* with $n$.
Furthermore, what the training does on a neural network (for time series forecasting) is solving the optimization problem

$$\min_{\boldsymbol{\theta} \in \boldsymbol{\theta}} \sum_{t=0}^{T} (\boldsymbol{f}(\boldsymbol{\theta}; \mathbf{x}^t) - \mathbf{y}^t)^2, \tag{1.22}$$

in the case of nonlinear least squares, where the output $\mathbf{y}_t$ is known at training time (it is contained in the dataset used for training) and $T$ is the number of time steps of the time series used for training. The function to minimize is called the *loss*

*function* and it is generally a nonlinear non-convex function, so this optimization problem is non-convex, which implies that the optimum found during the training might be a **local optimum**.
Finally, a form of regularization is usually performed during training to avoid overfitting, i.e. adapting too much to the training set, losing the ability to generalize to new data. This is done by modifying the minimization problem this way:

$$\min_{\boldsymbol{\theta}\in\boldsymbol{\theta}} \sum_{t=0}^{T} (\boldsymbol{f}(\boldsymbol{\theta}; \mathbf{x}^t) - \mathbf{y}^t)^2 + \lambda \|\mathbf{R}\boldsymbol{\theta}\|_2^2, \tag{1.23}$$

where $\lambda$ is the regularization factor and $\mathbf{R} \succeq \mathbf{0}$ is a square weighting matrix, usually chosen equal to the identity matrix. The higher $\lambda$, the smaller $\|\boldsymbol{\theta}\|_2$ and the stronger the regularization. This form of regularization is called Tikhonov regularization.

# Part II

# Design of robust observers with neural networks and linear systems examples

# Chapter 2

# Design of robust DVSs

In the previous part, the difference between the two-step and the one-step filter design procedures was explained, considering a situation where the equations of a system $S$ are unknown. In both cases, since there are unavoidable parametric uncertainties, it is important that the resulting filter is *robust* as the uncertainties on $S$ vary in a certain range, or, in other words, the filter estimation error variance must be as low as possible for every resulting $S$ in the considered range of uncertainties. Let us consider, for instance, a multiplicative uncertainty model, where $\boldsymbol{\xi} \in \mathbb{R}^{n_\xi}$ is the vector of parameters of $S$ that are subject to uncertainty (so $\boldsymbol{\xi}_{nom}$ is the vector of nominal parameters). Let $\boldsymbol{\Omega}_{min}$, $\boldsymbol{\Omega}_{max}$ be two diagonal weight matrices $\boldsymbol{\Omega}_{min}, \boldsymbol{\Omega}_{max} \in \mathbb{R}^{n_\xi, n_\xi}$ such that $\boldsymbol{\xi}_{max} = \boldsymbol{\Omega}_{max}\boldsymbol{\xi}_{nom}$, $\boldsymbol{\xi}_{min} = \boldsymbol{\Omega}_{min}\boldsymbol{\xi}_{min}$ (observe that choosing $\boldsymbol{\Omega} = \mathbf{I}$ we have $\boldsymbol{\Omega}\boldsymbol{\xi}_{nom} = \boldsymbol{\xi}_{nom}$). Note that the entries $(i, i)$ of $\boldsymbol{\Omega}$ are the eigenvalues $\lambda_i(\boldsymbol{\Omega})$. Let now $S_{\boldsymbol{\Omega}}$ be the resulting system when the perturbations associated to the weight matrix $\boldsymbol{\Omega}$ are applied to the parameters. We want the filter to be as accurate as possible for all $S_{\boldsymbol{\Omega}}$ with $\boldsymbol{\Omega}$ such that $\lambda_i(\boldsymbol{\Omega}_{min}) \leq \lambda_i(\boldsymbol{\Omega}) \leq \lambda_i(\boldsymbol{\Omega}_{max}),\ i = 1, \ldots, n_\xi$.
The two-step procedure consists of identifying a $S_{\boldsymbol{\Omega}_{nom}}$ from a set of measurements and designing a filter $\hat{K}$ that is optimal for $S_{\boldsymbol{\Omega}_{nom}}$. In the one-step procedure, instead, the measurements coming from $S_{\boldsymbol{\Omega}_{min}}$ and $S_{\boldsymbol{\Omega}_{max}}$ are used to directly obtain a filter that works in that range (the idea behind this procedure is to consider $S_{\boldsymbol{\Omega}_{min}}$ and $S_{\boldsymbol{\Omega}_{max}}$ as the limit cases on which the filter must still work). As is illustrated in the next chapters, this allows for more robustness, as the second filter (the DVS) has low variance in the whole range, whereas the performance of the first filter $\hat{K}$, instead, generally deteriorates evidently as the true parameters are distant from the nominal ones.
An important part for an accurate DVS design is the collection of the data to be used for training. Let us assume that, following the same notation of the previous part, we can measure a corrupted output $\mathbf{y}$ and we want to estimate a quantity $\mathbf{z}$ (that can be a linear or nonlinear combination of the states, that in turn may be

measurable or unmeasurable). Two phases can be distinguished: a training phase and a runtime phase. The training phase is where the data is collected to train the DVS; in this phase, the quantity $\mathbf{z}$ is assumed to be measurable or simulable exactly. The runtime phase, instead, is where the DVS is actually used to estimate $\mathbf{z}$, which is now unmeasurable. Now, if the system has a deterministic input $\mathbf{u}$ as well, a dataset

$$D_V := \{(\mathbf{u}(k), \mathbf{y}(k)),\ \mathbf{z}(k) \mid k = 1, \ldots, \frac{T}{T_s}\}, \tag{2.1}$$

where $T$ is the time length considered and $T_s$ is the sampling time. Since multiple experiments are performed to collect the data, then we can write

$$D_V := \{\{(\mathbf{u}_i(k), \mathbf{y}_i(k)),\ \mathbf{z}_i(k) \mid k = 1, \ldots, \frac{T}{T_s}\} \mid i = 1, \ldots, n_e\}, \tag{2.2}$$

where $n_e$ is the number of experiments. If a deterministic input $u$ is not present, instead, the dataset becomes

$$D_V := \{\{\mathbf{y}_i(k),\ \mathbf{z}_i(k) \mid k = 1, \ldots, \frac{T}{T_s}\} \mid i = 1, \ldots, n_e\}, \tag{2.3}$$

The parameters $T$ and $T_s$ are chosen so to observe the dynamics of the system long enough and to catch the relevant frequencies. Note that the experiments are to be performed when the system is in *open-loop*.
If $\mathbf{u}$ is present, a random input $\mathbf{u}$ has to be given to $S$ during the experiment (following for instance the Uniform distribution). If it is already known what the range of $\mathbf{u}$ will be at runtime (for instance due to the saturation of the actuators), then in the training phase $u$ must be a random input in that range.
The experiments in this thesis are performed half on $S_{\mathbf{\Omega}_{max}}$ and half on $S_{\mathbf{\Omega}_{min}}$, and the initial conditions of the system states vary at each experiment. After the dataset is collected, the DVS is trained on it. The architecture of the DVS is chosen prior to the training, and the choice includes the number of hidden units, the nonlinear function, the regularization factor etc. If there are multiple output variables to be predicted, it's better to design a DVS for each output, so that each DVS performs the optimal prediction for its related output.
As explained in the previous part, the DVS can be based on any form of linear or nonlinear approximation (linear functions, polynomials, etc.). The DVS method is the same with any choice, and the only difference lies in the structure of the approximation and, therefore, in the inference process. The DVSs designed in this thesis are based on neural networks. In the next chapter, the case of linear systems is considered.

# Chapter 3

# Robust DVSs for linear systems

In the case of linear systems, the performance of the filter obtained with the two-step approach noticeably decreases when $S_{\mathbf{\Omega}} \neq S_{\mathbf{\Omega}_{nom}}$ especially when $\hat{K}$ is the current Kalman filter (that is, however, the minimum-variance filter on $S_{\mathbf{\Omega}_{nom}}$. The situation is even worse when $\hat{K}$ is the delayed Kalman filter (see Appendix). To overcome this problem, other linear robust filters are found in the literature, based for instance on covariance matrix upper bounds ([3]) or on Linear Matrix Inequalities ([4]). In the following sections, two examples of linear systems are therefore considered, a DVS is designed for each example, and its robustness is compared to the one of a current Kalman filter (CKF), a delayed Kalman filter (DKF), and the filters of the papers cited above (that in this thesis are referred to as Shaked filter ([3]) and Duan filter ([4]). It is then shown that a robust DVS can be nonlinear even for linear systems.

## 3.1 Comparison with the Shaked filter

In [3] a robust discrete-time minimum variance observer is proposed for the following uncertain linear system:

$$\mathbf{x}_{k+1} = \begin{bmatrix} 0 & -0.5 \\ 1 & 1+\delta \end{bmatrix} \mathbf{x}_k + \begin{bmatrix} -6 \\ 1 \end{bmatrix} w_k, |\delta| < 0.3 \tag{3.1}$$

with the measurement

$$y_k = \begin{bmatrix} -100 & 10 \end{bmatrix} \mathbf{x}_k + \nu_k. \tag{3.2}$$

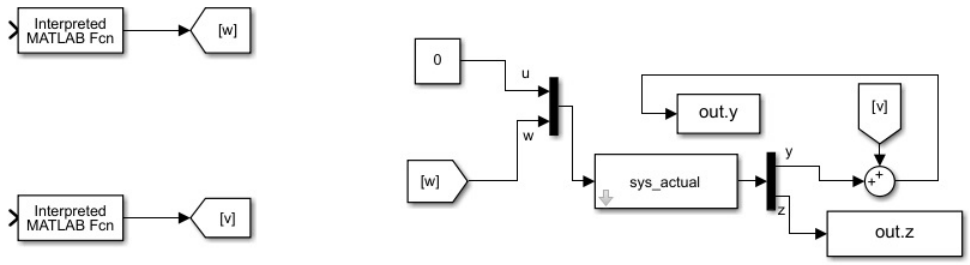
The process noise $w_k \in \mathbb{R}$ and the output noise $\nu_k \in \mathbb{R}$ are such that:

$$\mathbb{E}(w_k) = 0$$
$$\mathbb{E}(\nu_k) = 0$$
$$\mathbb{E}(\begin{bmatrix} w_k \\ \nu_k \end{bmatrix} \begin{bmatrix} w_l^T & \nu_l^T \end{bmatrix}) = \begin{cases} \mathbf{I}_2 & k = l \\ \mathbf{0} & k \neq l \end{cases}$$

We want to estimate $z_k = \begin{bmatrix} 1 & 0 \end{bmatrix} \mathbf{x}_k$. In the paper, the $H_2$ filtering problem is considered, where the objective is finding a minimum-variance filter that minimizes the variance of the estimation error $\mathbb{E}((z - \hat{z})^T (z - \hat{z}))$. A minimum-variance filter is obtained with the following state equation:

$$\hat{\mathbf{x}}_{k+1} = \begin{bmatrix} 0 & -0.5861 \\ 1 & 1.1895 \end{bmatrix} \hat{\mathbf{x}}_k + \begin{bmatrix} -0.0053 \\ 0.0016 \end{bmatrix} (y_k - \begin{bmatrix} -100 & 10 \end{bmatrix} \hat{\mathbf{x}}_k) \tag{3.3}$$

Figure 3.1. Simulink scheme of the system (Shaked case)



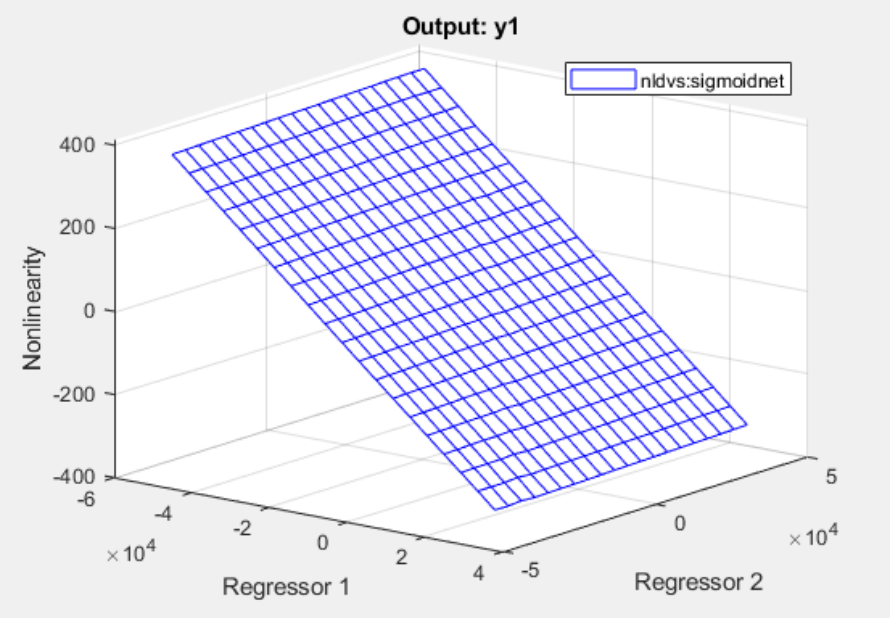

With $\delta = 0.5$, however, the system becomes unstable, and the proposed filter does not estimate correctly the status. In the following, it is shown how a well-designed DVS not only outperforms both the Kalman filter and the Shaked filter when $\delta \neq 0$, but also it correctly estimates the state even when $\delta = 0.5$ and the system is unstable. To obtain the DVS, the open-loop system is simulated for $1000s$ ten times with $\delta = 0.5$ and ten times with $\delta = -0.5$, always with $w_k, \nu_k \sim \mathcal{N}(\mu_{nom}, \sigma^2_{nom})$, where $\mu_{nom} = 0$, $\sigma^2_{nom} = 1$. The initial state $x_0$ changes each time, according to the uniform distribution between $-5$ and $+5$. We then merge the twenty experiments and use the resulting dataset for training. In Figure 3.1 the Simulink scheme of the system is shown, where there are two blocks that generate the random noise $w, \nu$ and two output blocks that gather the $y$ and $z$ data. *sys_actual* is the block representing the LTI system with the perturbation.

To obtain the DVS, the *nlarx* command in the System Identification toolbox is used. The filter is built considering:

- the corrupted system output $y$ as the filter input $u_1$
- the real $z$ as the filter output $y_1$
- a Sigmoid network with 5 units
- a sampling time $T_s = 1s$
- a regularization factor $\lambda = 1e - 2$
- the values $u_1(t), u_1(t-1)$ as regressors

After the training, we obtain an average training accuracy of 95.4%.

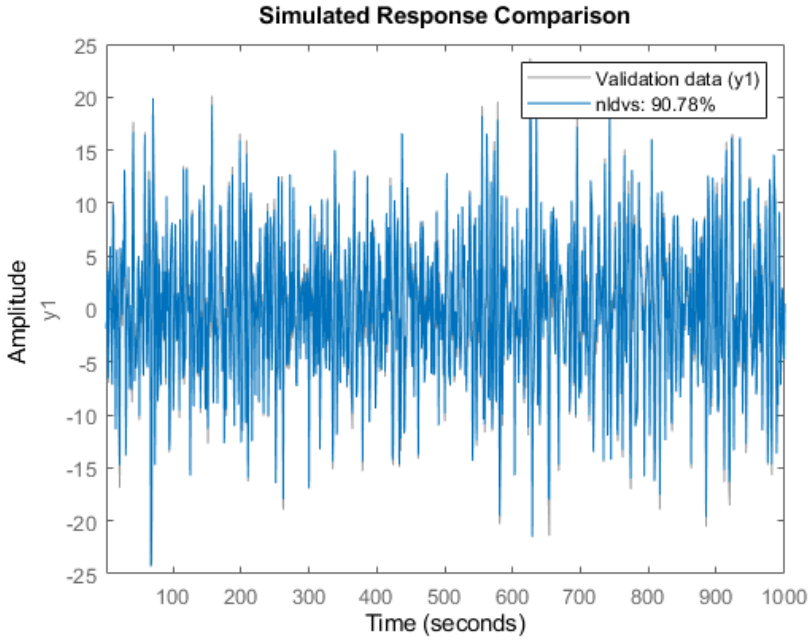
Figure 3.2. Plot of the nonlinearity of the DVS (Shaked case)




In Figure 3.2 the function $y_1(u_1(t), u_1(t-1))$ obtained is shown, and its almost linear nature is clear, since it resembles a plane, which is compatible to the linear nature of the system.
To obtain the validation accuracy, the system is simulated again and the system output $y$ is given as input to the filter; the filter output is then compared with $z$. In Figures 3.3, 3.4 and 3.5 the validation accuracy is shown. The higher accuracy when $\delta = -0.5, +0.5$ is due to the fact that the experiments used for training were performed with those values of $\delta$.

The variance of the estimation error is then measured, using (1) the current Kalman filter, (2) the delayed Kalman filter, (3) the Shaked filter, (4) the DVS.

Figure 3.3. Validation accuracy with $\delta = 0$ (Shaked case)



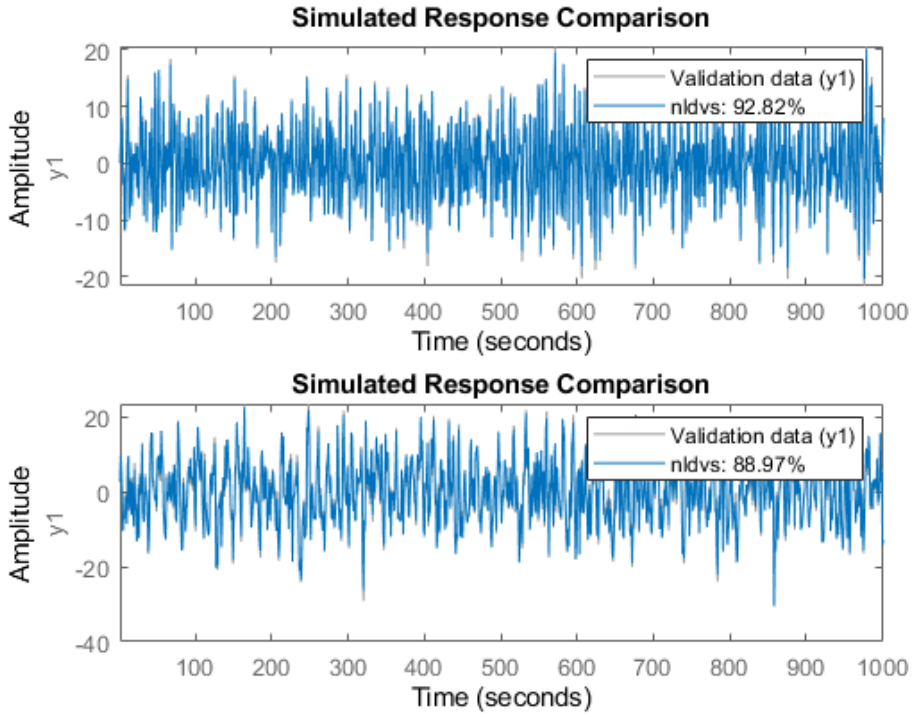

First, the performance is measured varying $\delta$ and with $w_k, \nu_k \sim \mathcal{N}(\mu, \sigma^2)$, keeping $\mu = \mu_{nom}$, $\sigma^2 = \sigma^2_{nom}$. Ten simulations are performed where the initial conditions are randomly chosen according to the uniform distribution between $-5$ and $+5$, and the average variance is calculated.

| $\delta$ | Current Kalman | Delayed Kalman | Shaked | NL DVS |
|---|---|---|---|---|
| 0 | **0.000245** | 35.9485 | 44.3469 | 0.4515 |
| +0.3 | 57.8721 | 1523.3464 | 52.3918 | **0.8852** |
| −0.3 | 4.4321 | 142.459 | 47.9492 | **0.2943** |
| +0.5 | 6.978e+06 | 1.739e+08 | 6.176e+03 | **1.237** |
| −0.5 | 8.8321 | 251.4313 | 48.3995 | **0.2254** |

Table 3.1. Performances varying only $\delta$ (Shaked case)

As we can see from Table 3.1, the current Kalman filter is the best only on the nominal system. On the perturbed ones, however, the DVS is noticeably the best, even when $\delta = 0.5$.
Another type of robustness is then tested other than the one to parametric uncertainty. Using the same DVS found earlier, its accuracy is tested against uncertainty

Figure 3.4. Validation accuracy with $\delta = -0.3, +0.3$ (Shaked case)




on the noise structure. Therefore, keeping $\delta = 0$, we vary $\sigma_w^2$ and $\sigma_\nu^2$.

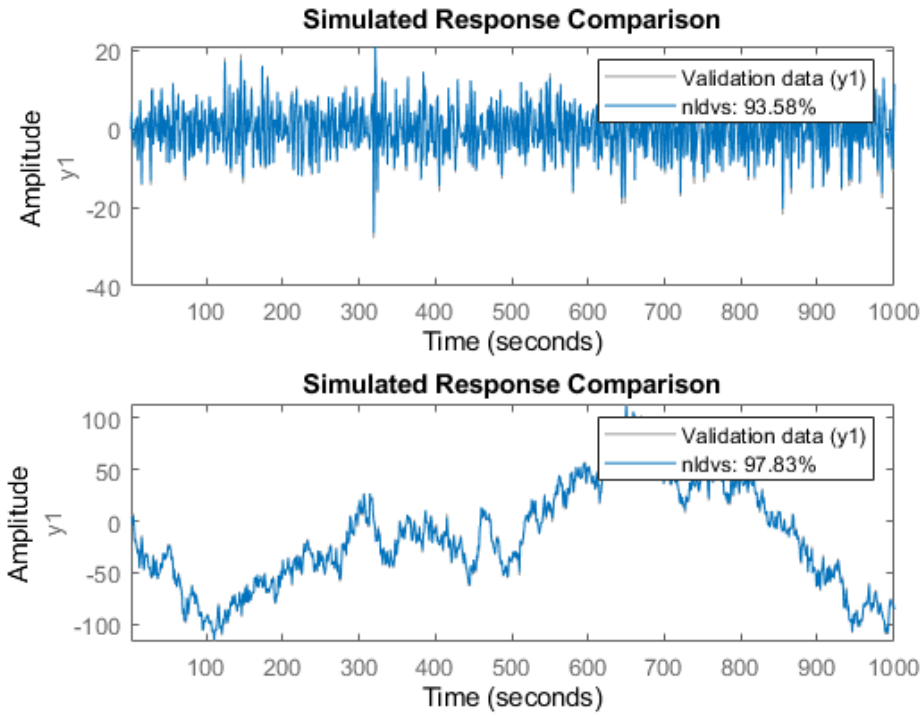
| $\sigma_w^2$ | $\sigma_\nu^2$ | Current Kalman | Delayed Kalman | Shaked | NL DVS |
|---|---|---|---|---|---|
| 1 | 10 | **0.0022009** | 35.9008 | 45.8755 | 0.43558 |
| 10 | 1 | **0.0010877** | 356.7499 | 448.6862 | 10.3115 |
| 10 | 10 | **0.0029875** | 372.5341 | 468.9653 | 10.6136 |

Table 3.2. Performances varying only $\sigma_w^2$ and $\sigma_\nu^2$ (Shaked case)

As we can see from Table 3.2, the current Kalman filter is the best and this is coherent with the fact that the system is the nominal one. However, the DVS still outperforms both the delayed Kalman filter and the Shaked filter. Also, the noise $w$ appears to be more impactful than $\nu$.
Finally, we test the DVS when the noises do not even follow the gaussian distribution, but the uniform one, varying the interval of the distribution.

Based on the results shown in Table 3.3, also in this case the current Kalman filter is the best and the DVS outperforms both the delayed Kalman filter and

Figure 3.5. Validation accuracy with $\delta = -0.5, +0.5$ (Shaked case)



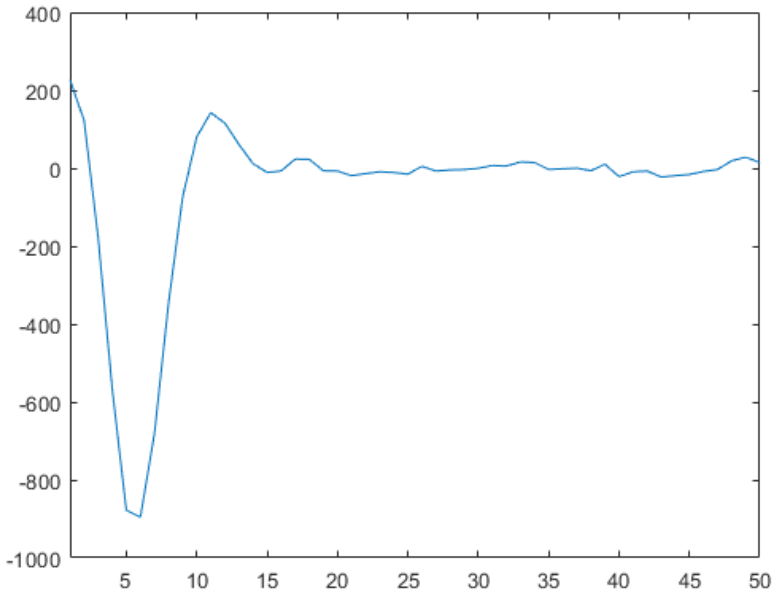

| $[a, b]$ | Current Kalman | Delayed Kalman | Shaked | NL DVS |
|---|---|---|---|---|
| [−1,1] | **8.8387e-05** | 11.9941 | 15.0644 | 0.14007 |
| [−2,2] | **0.00032737** | 48.3574 | 61.3474 | 0.61263 |
| [−10,10] | **0.0065642** | 1212.1872 | 1532.4783 | 37.2973 |

Table 3.3. Performances without uncertainty but with uniformly distributed noises (Shaked case)

the Shaked filter. The decrease in accuracy of the DVS is due to the fact that, at training time, only Gaussian white noises were used. In an application where the real noise variance or distribution is unknown, the training should be done using different variances and distributions.
Finally, we want to see how robust is the observer when the initial conditions are different than the ones used for training (that were uniform between $-5$ and 5). For instance, if we set $\mathbf{x}_0 = [1000; 1000]$, with $\delta = 0$ and $w, \nu$ Gaussian white noises, we obtain an average variance equal to 81.7944. As shown in Figure 3.6,

where the estimation error in the first 50 seconds is plotted, this is due to the fact that, at the beginning, the error is very high, even though it converges to zero after a few seconds. Indeed, the average error variance ignoring the first 10 seconds is equal to 0.5552.

Figure 3.6. Estimation error with $x_0 = [1000; 1000]$ (Shaked case)




## 3.2 Comparison with the Duan filter

In [4] another robust filter design procedure is proposed by means of linear matrix inequalities (LMI). In particular, the example number 2 on the paper is used to compare the performance of the Duan filter against a DVS. This example is very similar to the one in [3], but the objective is different. Given the system described by the following matrices:

$$\mathbf{A} = \begin{bmatrix} 0 & -0.5 \\ 1 & 1+\delta \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} -6 & 0 \\ 1 & 0 \end{bmatrix},$$

$$\mathbf{C} = \begin{bmatrix} -100 & 10 \end{bmatrix}, \quad \mathbf{D} = \begin{bmatrix} 0 & 1 \end{bmatrix}, \quad \mathbf{L} = \begin{bmatrix} 1 & 0 \end{bmatrix}, \quad |\delta| \leq 0.45$$

the $H_\infty$ problem is here considered instead: given a scalar $\gamma_\infty$, the objective is finding a stable filter such that

$$\|z - z\|_2 < \gamma_\infty \|\mathbf{w}\|_2 \tag{3.4}$$

for all $\mathbf{w}_k \in l_2[0, +\infty)$. As a performance index, then, the value $\frac{\|z-z\|_2}{\|\mathbf{w}\|_2}$ is considered (the value $\|\mathbf{w}\|_2$ is assumed to be known at simulation time). In the paper, a filter with the following matrices is found:

$$\mathbf{A}_F = \begin{bmatrix} 1.9248 & 0.6322 \\ -6.1254 & -1.9575 \end{bmatrix}, \quad \mathbf{B}_F = \begin{bmatrix} 0.0038 \\ 0.0071 \end{bmatrix} \tag{3.5}$$

$$\mathbf{L}_F = \begin{bmatrix} -0.1590 & -0.0605 \end{bmatrix}, \quad \mathbf{D}_F = \begin{bmatrix} -0.0078 \end{bmatrix} \tag{3.6}$$

and we want to test its performance against a DVS. As before, 10 simulations with $\delta = 0.5$ and 10 with $\delta = -0.5$ are merged and used for training (each one with uniformly distributed initial conditions between $-5$ and 5). The scheme in Figure 3.7 is used to perform the experiments. Even though it seems that there is no output disturbance, the input noise $\mathbf{w}$ is a vector with two components and, by looking at the matrices $\mathbf{B}$ and $\mathbf{D}$, it is evident that the first components affects $\mathbf{x}$ whereas the second affects $y$. The same configuration as before for the DVS is used, i.e. we consider

- the system output $y$ as the filter input $u_1$
- the real $z$ as the filter output $y_1$
- a Sigmoid network with 5 units
- a sampling time $T_s = 1s$
- a regularization factor $\lambda = 1e - 2$
- the values $u_1(t), u_1(t-1)$ as regressors

In Figure 3.8 the output function $y_1(u_1(t), u_1(t-1))$ is plotted, and once again is coherent with the linearity of the system since it is an almost perfect plane.

The validation accuracy is then tested, first on the nominal system (Figure 3.9), then with $\delta = -0.45, +0.45$ (Figure 3.10), and finally with $\delta = -0.5, +0.5$ (Figure 3.11).

The higher accuracy with $\delta = -0.5, +0.5$ is due to the fact that the training was performed with those values of $\delta$, but the accuracy with $\delta = 0, -0.45, +0.45$ is comparable. Finally, the obtained DVS is used to compare its performance (the $H_\infty$ index) to the one of the Kalman filter and the Duan filter, varying $\delta$. 10

Figure 3.7. Simulink scheme of the system (Duan)

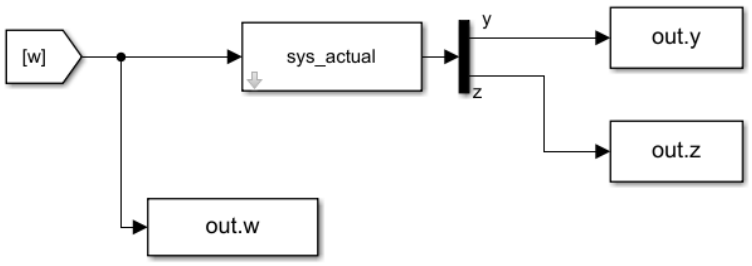
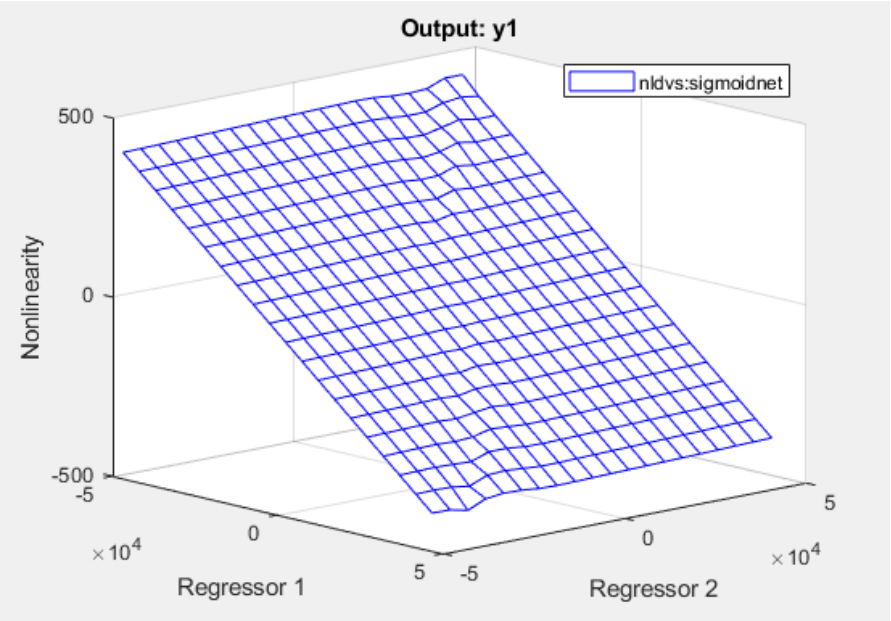



Figure 3.8. Plot of the nonlinearity of the DVS (Duan case)

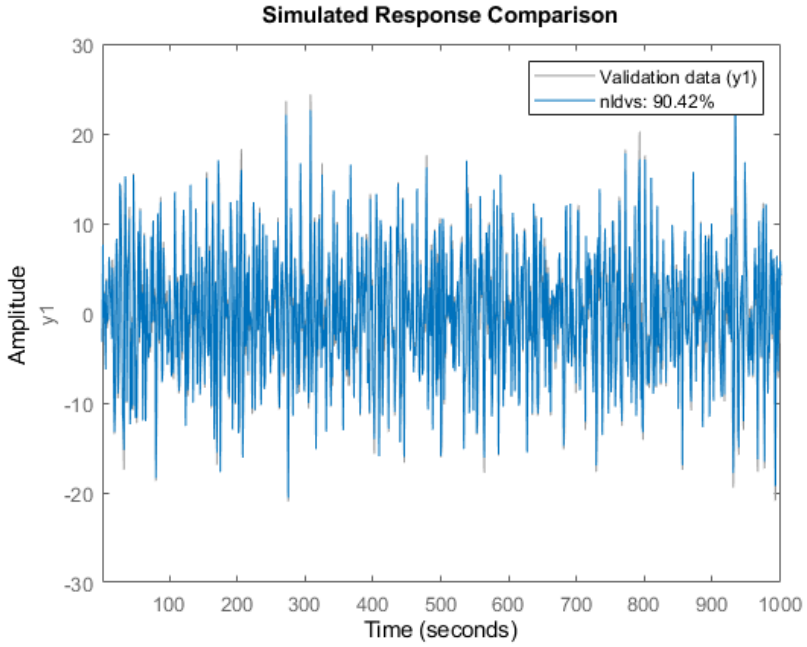



simulations are done for 1000 seconds, and also the noise $w$ is collected so that its norm can be calculated.

The $H_\infty$ index is calculated for each simulation and the average is shown in Table 3.4.

It is evident that, excluding the nominal case where the current Kalman filter is the best, the DVS outperforms all the other filters, even when the system is unstable ($\delta = +0.5$).

Figure 3.9. Validation accuracy with $\delta = 0$ (Duan case)




| $\delta$ | Current Kalman | Delayed Kalman | Duan | DVS |
|---|---|---|---|---|
| 0 | **0.16186** | 6.0563 | 1.2643 | 0.67028 |
| $-0.45$ | 3.5484 | 18.8207 | 1.2612 | **0.50454** |
| 0.45 | 64.7371 | 323.5909 | 1.3035 | **1.2369** |
| $-0.5$ | 4.3688 | 22.7127 | 1.2495 | **0.47786** |
| $+0.5$ | 2628.0843 | 13115.0147 | 1.5685 | **1.1169** |

Table 3.4. Performances varying only $\delta$ (Duan case)

Figure 3.10. Validation accuracy with $\delta = -0.45, +0.45$ (Duan case)

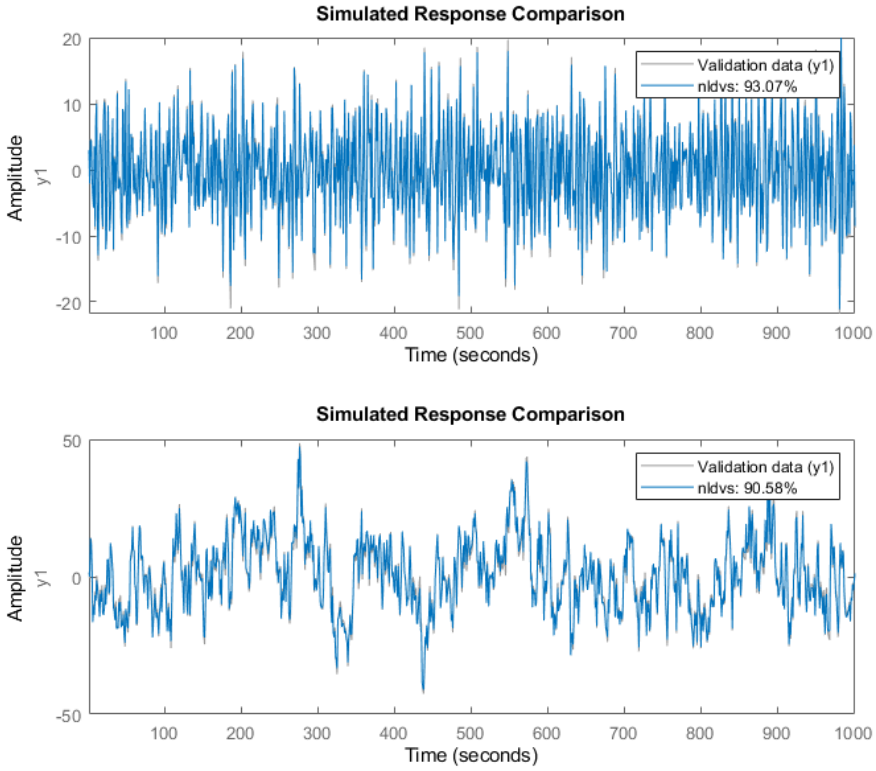
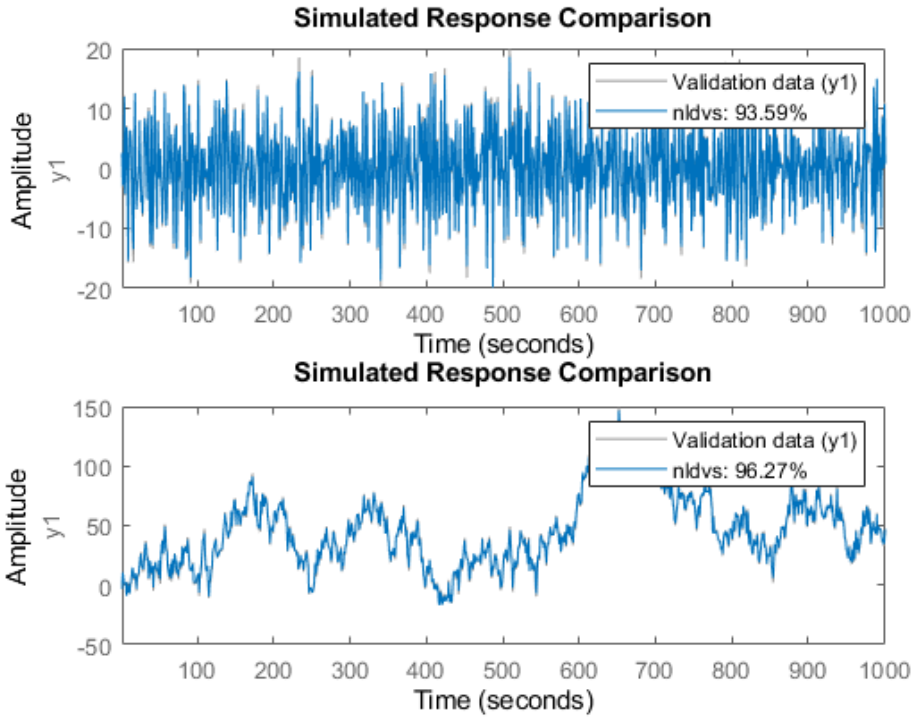


Figure 3.11. Validation accuracy with $\delta = -0.5, +0.5$ (Duan case)



# Part III

# Spacecraft attitude robust control with neural-network-based observers

# Chapter 4

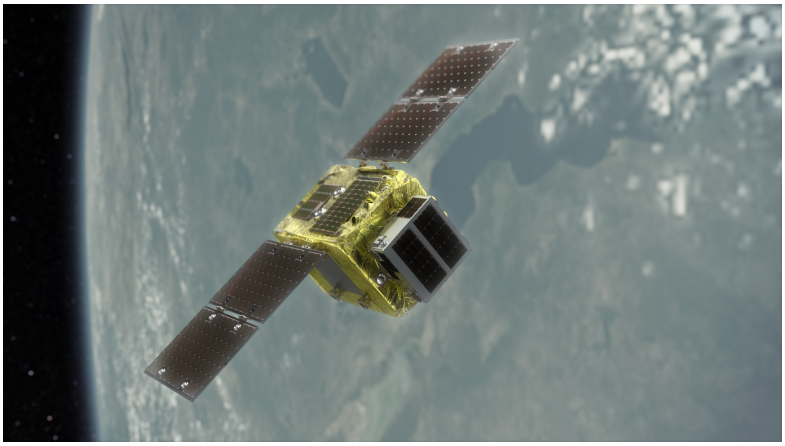
# Spacecraft attitude control

## 4.1 Introduction

It is of fundamental importance to control the attitude of a spacecraft, which can be a space station orbiting around a planet, a satellite to be pointed towards a target, a space vehicle etc.
In the case of satellites, the orientation must be usually controlled with respect to either a point on Earth (Earth pointing satellites) or a celestial frame (inertial pointing satellites). However, controlling a nonlinear system is a delicate task that requires high accuracy, and high robustness is needed whenever an exact model of the system cannot be obtained, or when the system itself can change during a mission. The second scenario is considered in this thesis, where an example of a satellite used to hook space debris is explored. The problem of space debris is becoming more and more important as time goes by. Particularly in Earth orbit, indeed, there is a strong presence of derelict spacecraft, which is in turn subject to disintegration and collisions that generate more waste. Furthermore, even the unburned particles of rocket motors and the solidified liquids expelled from spacecrafts contribute to the high amount of debris that represents a risk to spacecrafts and an environmental problem. Space companies are developing solutions to this problem, by designing spacecrafts (satellites, for instance) that are able to track and hook objects in space. An example is the ELSA-d (End-of-Life Service by Astroscale) satellite, designed by the japanese company Astroscale and shown in Figure 4.1. However, after a spacecraft hooks a tensor, its inertia matrix can vary substantially, depending on the object geometry and mass and from the position of the hook with respect to the spacecraft axes. Generally speaking, we can imagine that the inertia matrix can even triplicate, so the ACS must work well also in that case and robustness is needed. In the example explored in this thesis, it is supposed that the angular speed can not be measured. The industry standard

in this case would be to use an extended Kalman filter to estimate it, but it is not guaranteed to be the minimum-variance filter, neither the most robust one. A DVS is therefore deployed and its performance is compared to the one of the EKF, both in closed-loop and in open-loop. However, the system must first be modeled so that the simulation can be done; in the following sections, the spacecraft dynamics are described to derive an attitude control system, as in [7].

Figure 4.1. A rendering of ELSA-d (Author: Astroscale, CC-BY-SA)



## 4.2 Rotations

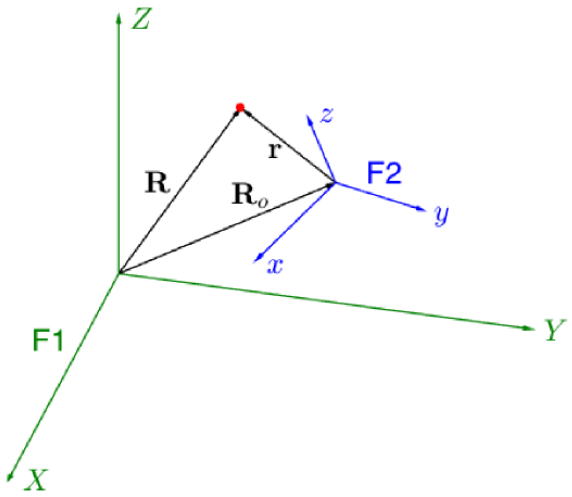
Let us first start by defining the concept of **frame of reference**. An *orthogonal* frame of reference $\mathfrak{R} = \{O, \mathbf{i,j,k}\}$ is formed by an origin $O$ and a set of three mutually orthogonal versors $\{\mathbf{i,j,k}\}$ with origin in $O$. We can dinstiguish three kinds of RFs:

- body frames: the origin and the axes are defined by points of a rigid body, and the body Center of Mass (CoM) is taken as the origin;
- trajectory frames: the axes are aligned with the three instantaneous directions of the body trajectory in space, and the body CoM is taken as the origin;
- celestial frames: the origin and the axis are defined by points and directions in the universe.

Let us now consider two RFs:

- $F1 = \{O_1, \mathbf{I,J,K}\}$ with axes $X, Y, Z$,

Figure 4.2. Representing a point with respect to two different reference frames




- $F2 = \{O_2, \mathbf{i},\mathbf{j},\mathbf{k}\}$ with axes $x, y, z$,

where $\mathbf{R}_0 = X_0\mathbf{I} + Y_0\mathbf{J} + Z_0\mathbf{K}$ is the position of the origin of F2 w.r.t. F1. If we have a point in space expressed as $\mathbf{R} = X\mathbf{I} + Y\mathbf{J} + Z\mathbf{K}$ w.r.t. F1 and as $\mathbf{r} = x\mathbf{i} + y\mathbf{j} + z\mathbf{k}$ w.r.t. F2, we can write

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} X_0 \\ Y_0 \\ Z_0 \end{bmatrix} + \mathbf{T} \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \mathbf{T} := \begin{bmatrix} \mathbf{I}\cdot\mathbf{i} & \mathbf{I}\cdot\mathbf{j} & \mathbf{I}\cdot\mathbf{k} \\ \mathbf{J}\cdot\mathbf{i} & \mathbf{J}\cdot\mathbf{j} & \mathbf{J}\cdot\mathbf{k} \\ \mathbf{K}\cdot\mathbf{i} & \mathbf{K}\cdot\mathbf{j} & \mathbf{K}\cdot\mathbf{k} \end{bmatrix},$$

where $\mathbf{T}$ is the *direction cosine matrix* (DCM). Since translations and rotations can be treated independently, and we are more interested in rotations, it is assumed $\mathbf{R}_0 = 0$, so that

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \mathbf{T} \begin{bmatrix} x \\ y \\ z \end{bmatrix},$$

and we can see that $\mathbf{T}$ is the transformation of the coordinates from F2 to F1. There is also another interpretation of DCMs: instead of a transformation from the coordinates in F2 to the ones in F1, $\mathbf{Tv}$ can be seen as the rotation of $\mathbf{v}$ from the orientation of F1 to the one of F2, but remaining in the fixed frame F1.

### 4.2.1 Euler angles

Considering a fixed frame of reference with axes $(X, Y, Z)$, we can define three elementary rotation matrices, representing:

- a rotation by an angle $\phi$ about X

$$\mathbf{T}_1(\phi) := \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix}$$

- a rotation by an angle $\theta$ about Y

$$\mathbf{T}_2(\theta) := \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}$$

- a rotation by an angle $\psi$ about Z

$$\mathbf{T}_3(\psi) := \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Any rotation can be expressed by a composition of the elementary rotations, that mathematically corresponds to a product of $\mathbf{T}_1, \mathbf{T}_2, \mathbf{T}_3$. Rotations can be either **extrinsic** or **intrinsic**. To be more clear, if we have two RFs, one fixed with axes $X, Y, Z$ and one rotating with axes $x, y, z$, extrinsic rotations are about the axes of the fixed frame, whereas intrinsic rotations are about the axes of the rotating one. For instance, the matrix $\mathbf{T}_1(\phi)\mathbf{T}_2(\theta)\mathbf{T}_3(\psi)$ can be seen as an extrinsic rotation about Z (by $\psi$), then about Y (by $\theta$) and finally about X (by $\phi$), or as an intrinsic rotation about x (by $\phi$), then about y' (by $\theta$) and finally about z" (by $\psi$).

### 4.2.2 Quaternions

According to Euler's rotation theorem, any displacement of a rigid body such that one point is fixed is equivalent to a rotation about an axis passing through the fixed point. The axis **u** is the eigenvector corresponding to the eigenvalue 1 of the rotation matrix.
This means that we can describe any 3D rotation with four variables, one representing the angle and three representing the axis. We now introduce the *Euler parameters*: a rotation of an angle $\beta$ about an axis defined by the versor

$\mathbf{u} = u_1\mathbf{i} + u_2\mathbf{j} + u_3\mathbf{k}$ can be described by the following four parameters:

$$q_0 := \cos\frac{\beta}{2}$$
$$q_1 := u_1 \sin\frac{\beta}{2}$$
$$q_2 := u_2 \sin\frac{\beta}{2}$$
$$q_3 := u_3 \sin\frac{\beta}{2}$$

These parameters form a **quaternion**. Quaternions are elements of a four-dimensional linear vector space with basis $\{1, \mathbf{i}, \mathbf{j}, \mathbf{k}\}$. A quaternion $\mathfrak{q}$ can be written as

$$\mathfrak{q} = q_0 + q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{j} = q_0 + \mathbf{q} = (q_0, q_1, q_2, q_3),$$

where $q_0$ is called the *real part* and $\mathbf{q}$ is called the *imaginary part*.

Any vector $\mathbf{r} = (x, y, z)$ in the 3D space can be seen as a quaternion with null real part, i.e. $(0, \mathbf{r})$. Let us now consider the situation where we want to rotate $\mathbf{r}$ about the axis $\mathbf{u}$ by an angle $\alpha$ and let $\mathfrak{q} = (\cos\frac{\alpha}{2}, u_1 \sin\frac{\alpha}{2}, u_2 \sin\frac{\alpha}{2}, u_3 \sin\frac{\alpha}{2})$ be the quaternion corresponding to the rotation. The vector $\mathbf{r}'$ resulting from the rotation can be calculated as

$$(0, \mathbf{r}') = \mathfrak{q} \otimes (0, \mathbf{r}) \otimes \mathfrak{q}^*,$$

where $\mathfrak{q}^*$ is the conjugate quaternion of $\mathfrak{q}$ defined as $(q_0, -\mathbf{q})$ and $\otimes$ is the Hamilton product:

$$\mathfrak{q} \otimes \mathfrak{p} = (q_0 p_0 - \mathbf{q} \cdot \mathbf{p}) + (q_0\mathbf{p} + p_0\mathbf{q} + \mathbf{q} \times \mathbf{p}).$$

It follows that a composition of rotations $\mathbf{T} = \mathbf{T}_1\mathbf{T}_2 \ldots \mathbf{T}_n$ corresponds to the quaternion $\mathfrak{q} = \mathfrak{q}_1 \otimes \mathfrak{q}_2 \otimes \cdots \otimes \mathfrak{q}_n$, where each $\mathfrak{q}_i$ corresponds to the rotation $\mathbf{T}_i$, and the inverse of a rotation defined by $\mathfrak{q}$ is $\mathfrak{q}^{-1} = \mathfrak{q}^*$.

## 4.3 Attitude kinematics

The first step towards obtaining a dynamic model for attitude control is studying the attitude *kinematics*. A satellite can be described as a rigid body moving respect to an inertial frame of reference, where this movement is given by a combination of

- a translation of the body center of mass,
- a rotation about an axis passing through the center of mass.

Applying a moment $\mathbf{M}$ to the body, an angular velocity $\mathbf{w}$ will be generated, according to the dynamic equations that will be described later. What is of interest in this section is the relationship between the angular velocity $\mathbf{w}$, generated by $\mathbf{M}$, and the resulting evolution of the orientation, described by the kinematic equations.
Let us consider two reference frames,

- an inertial Observer frame (OF) with versors $(\mathbf{i}_1, \mathbf{i}_2, \mathbf{i}_3)$ and axes $X, Y, Z$,
- a rotating Body frame (BF) with origin on the body CoM, versors $(\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3)$ and axes $x, y, z$; the rigid body rotates with angular velocity $\boldsymbol{\omega} = \omega_1\mathbf{b}_1 + \omega_2\mathbf{b}_2 + \omega_3\mathbf{b}_3$.

Since we are going to use the quaternions to describe the attitude, the goal will be to describe the time evolution of $\mathfrak{q}$, the rotation quaternion of the rigid body, in function of $\omega_1, \omega_2, \omega_3$.
Both $\mathfrak{q}$ and $\boldsymbol{\omega}$ are a function of time: $\mathfrak{q} \equiv \mathfrak{q}(t)$ and $\boldsymbol{\omega} \equiv \boldsymbol{\omega}(t)$. From time $t$ to time $t + \Delta t$, a rotation $\Delta\mathfrak{q}(t)$ happens, starting from the initial orientation $\mathfrak{q}(t)$, so that at time $t + \Delta t$ the new orientation is the composition of the two rotations, i.e.

$$\mathfrak{q}(t + \Delta t) = \mathfrak{q}(t) \otimes \Delta\mathfrak{q}(t).$$

To mathematically express $\Delta\mathfrak{q}(t)$, let $\mathbf{u}$ be the versor of the rotation axis, and let $\boldsymbol{\omega} = |\boldsymbol{\omega}|$ be the magnitude of the angular speed. We have that $\boldsymbol{\omega} = \boldsymbol{\omega}\mathbf{u}$. Also, for a small $\Delta t$, the rotation angle is $\boldsymbol{\omega}\Delta t$. This means that for a small $\Delta t$ we can write

$$\Delta\mathfrak{q} \approx \begin{bmatrix} \cos\frac{\omega\Delta t}{2} \\ \mathbf{u}\sin\frac{\omega\Delta t}{2} \end{bmatrix} \approx \begin{bmatrix} 1 \\ \mathbf{u}\frac{\omega\Delta t}{2} \end{bmatrix} = \begin{bmatrix} 1 \\ \frac{\omega\Delta t}{2} \end{bmatrix} \tag{4.1}$$

We can finally write the quaternion derivative as

$$\dot{\mathfrak{q}} = \lim_{\Delta t\to 0} \frac{\mathfrak{q} \otimes \Delta q - \mathfrak{q}}{\Delta t} = \lim_{\Delta t\to 0} \frac{\mathfrak{q} \otimes ((1, \boldsymbol{w}\Delta t/2) - (1, \mathbf{0}))}{\Delta t} =$$
$$= \lim_{\Delta t\to 0} \frac{\mathfrak{q} \otimes (0, \frac{\omega\Delta t}{2})}{\Delta t} = \frac{1}{2}\mathfrak{q} \otimes (0, \boldsymbol{\omega})$$

and, defining the matrix

$$\mathbf{Q} := \begin{bmatrix} -q_1 & -q_2 & -q_3 \\ q_0 & -q_3 & q_2 \\ q_3 & q_0 & -q_1 \\ -q_2 & q_1 & q_0 \end{bmatrix}, \tag{4.2}$$

this is equivalent to the following kinematic equation:

$$\dot{\mathfrak{q}} = f(\mathfrak{q}, \boldsymbol{\omega}) = \frac{1}{2}\mathbf{Q}\boldsymbol{\omega}. \tag{4.3}$$

## 4.4 Attitude dynamics

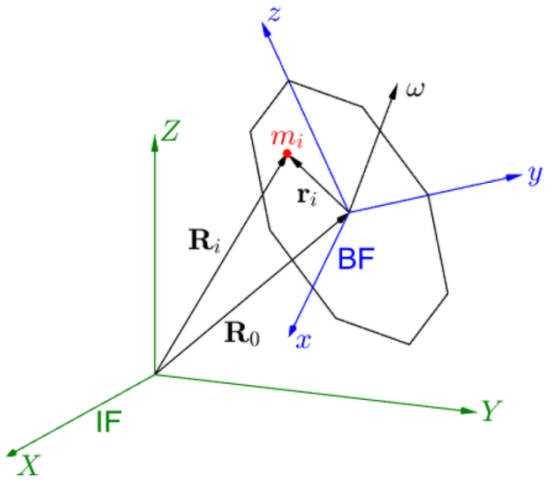
In the previous section, a dynamic relationship between the angular velocity $\boldsymbol{\omega}$ and the orientation quaternion $\mathfrak{q}$ was found. In this section it is instead described how to derive a dynamic relationship between the input moment $\mathbf{M} = M_1, M_2, M_3$ and the angular velocity $\boldsymbol{\omega}$.
Let us consider again two reference frames,

- an inertial Observer frame (OF) with versors $(\mathbf{i}_1, \mathbf{i}_2, \mathbf{i}_3)$ and axes $X, Y, Z$,
- a rotating Body frame (BF) with origin on the body CoM, versors $(\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3)$ and axes $x, y, z$; the rigid body rotates with angular velocity $\boldsymbol{\omega} = \omega_1\mathbf{b}_1 + \omega_2\mathbf{b}_2 + \omega_3\mathbf{b}_3$.

Now, considering a point of the body with mass $m_i$, we can see in Figure 4.3 that:

$$\mathbf{R}_0 = X_0\mathbf{i}_1 + Y_0\mathbf{i}_2 + Z_0\mathbf{i}_3 \tag{4.4}$$
$$\mathbf{R}_i = X\mathbf{i}_1 + Y\mathbf{i}_2 + Z\mathbf{i}_3 = \mathbf{R}_0 + \mathbf{r}_i \tag{4.5}$$
$$\dot{\mathbf{R}}_i = \dot{\mathbf{R}}_0 + \mathbf{R}_{iB} + \boldsymbol{\omega} \times \mathbf{r}_i \tag{4.6}$$
$$\mathbf{r}_i = x\mathbf{b}_1 + y\mathbf{b}_2 + z\mathbf{b}_3 \tag{4.7}$$
$$\dot{\mathbf{r}}_{iB} = \dot{x}\mathbf{b}_1 + \dot{y}\mathbf{b}_2 + \dot{z}\mathbf{b}_3 \tag{4.8}$$
$$\boldsymbol{\omega} = \omega_1\mathbf{b}_1 + \omega_2\mathbf{b}_2 + \omega_3\mathbf{b}_3 \tag{4.9}$$

Figure 4.3. Representing a body point with respect to two different reference frames




Now, the **angular momentum** of the point can be defined as

$$\mathbf{H}_i := \mathbf{r}_i \times m_i\dot{\mathbf{R}}_i = \mathbf{r}_i \times m_i(\dot{\mathbf{R}}_0 + \dot{\mathbf{r}}_{iB} + \boldsymbol{\omega} \times \mathbf{r}_i) \tag{4.10}$$

and, observing that $\dot{\mathbf{r}}_{iB} = 0$ since the body is rigid and there is no deformation, we have that

$$\mathbf{H}_i = \mathbf{r}_i \times m_i(\dot{\mathbf{R}}_0 + \boldsymbol{\omega} \times \mathbf{r}_i) = -\dot{\mathbf{R}}_0 \times m_i\mathbf{r}_i + \mathbf{r}_i \times m_i(\boldsymbol{\omega} \times \mathbf{r}_i). \tag{4.11}$$

For $m_i \to dm$, the angular momentum of the entire body is

$$\mathbf{H} = -\dot{\mathbf{R}}_0 \times \iiint_B \mathbf{r}\, dm + \iiint_B \mathbf{r} \times (\boldsymbol{\omega} \times \mathbf{r})\, dm \tag{4.12}$$

and, since the body frame has origin on the center of mass, so that $\iiint_B \mathbf{r}\, dm = 0$ by definition, we have that

$$\mathbf{H} = \iiint_B \mathbf{r} \times (\boldsymbol{\omega} \times \mathbf{r})\, dm \tag{4.13}$$

Performing the calculations and using the linearity of the integral operator, and being $\mathbf{r} = x\mathbf{b}_1 + y\mathbf{b}_2 + z\mathbf{b}_3 = (x, y, z)$ we can write $\mathbf{H}$ as $(H_1, H_2, H_3)$, where

$$H_1 = \iiint_B (y^2 + z^2)\omega_1\, dm - \iiint_B xy\omega_2\, dm - \iiint_B xz\omega_3\, dm \tag{4.14}$$

$$H_2 = -\iiint_B xy\omega_1\, dm + \iiint_B (x^2 + z^2)\omega_2\, dm - \iiint_B yz\omega_3\, dm \tag{4.15}$$

$$H_3 = -\iiint_B xz\omega_1\, dm - \iiint_B yz\omega_2\, dm + \iiint_B (x^2 + z^2)\omega_3\, dm \tag{4.16}$$

so that we can write

$$\mathbf{H} = \begin{bmatrix} J_{11} & J_{12} & J_{13} \\ J_{21} & J_{22} & J_{23} \\ J_{31} & J_{32} & J_{33} \end{bmatrix} \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix} = \mathbf{J}\boldsymbol{\omega}, \tag{4.17}$$

where

$$J_{11} = \iiint_B (y^2 + z^2)\, dm \tag{4.18}$$

$$J_{22} = \iiint_B (x^2 + z^2)\, dm \tag{4.19}$$

$$J_{33} = \iiint_B (x^2 + y^2)\, dm \tag{4.20}$$

are called the *moments of inertia* and

$$J_{12} = J_{21} = -\iiint_B xy\, dm \tag{4.21}$$

$$J_{13} = J_{31} = -\iiint_B xz\, dm \tag{4.22}$$

$$J_{23} = J_{32} = -\iiint_B yz\, dm \tag{4.23}$$

are called the *products of inertia*. The matrix $\mathbf{J}$ is symmetric and it is called the *inertia matrix*, or also *inertia tensor*.
It is known from linear algebra (the *spectral theorem*) that not only every symmetric matrix $\mathbf{J}'$ is always similar to a diagonal matrix $\mathbf{J}$, so that it exists an invertible matrix $\mathbf{M}$ such that $\mathbf{J} = \mathbf{M}^{-1}\mathbf{J}'\mathbf{M}$, but, since $\mathbf{J}'$ is symmetric, $\mathbf{M}$ is orthogonal, so we can always find a diagonal $\mathbf{J}$ starting from a non-diagonal $\mathbf{J}'$, given

$$\mathbf{J} = \mathbf{M}^T\mathbf{J}'\mathbf{M}. \tag{4.24}$$

This corresponds to rotating the body frame so that the three axes correspond to the *principal inertia axis*.
If a moment $\mathbf{M} = M_1\mathbf{b}_1 + M_2\mathbf{b}_2 + M_3\mathbf{b}_3$ is acting on the body, it is known by the second law of dynamics for a rotating body that

$$\dot{\mathbf{H}} = \mathbf{M}, \tag{4.25}$$

and, being $\dot{\mathbf{H}} = \mathbf{J}\boldsymbol{\omega} + \boldsymbol{\omega} \times \mathbf{H}$, the **Euler moment equation** is obtained:

$$\mathbf{J}\dot{\boldsymbol{\omega}} = \mathbf{M} - \boldsymbol{\omega} \times \mathbf{J}\boldsymbol{\omega}. \tag{4.26}$$

## 4.5 Attitude control

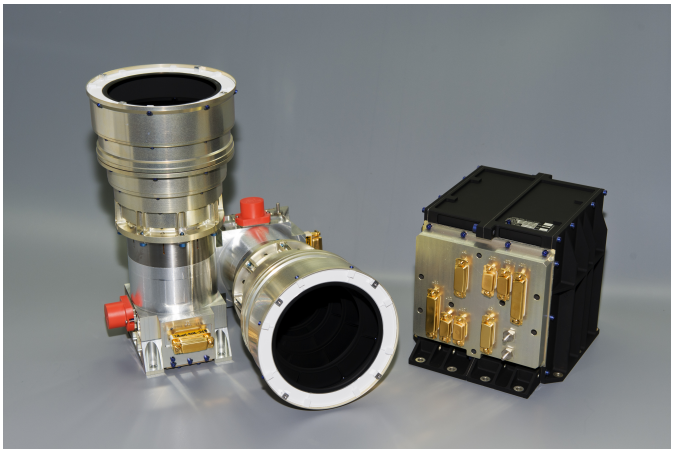
On a spacecraft, the Attitude Control System (ACS) is of fundamental importance to accurately control the orientation of the spacecraft during a mission. To achieve this goal, *sensors* are needed to determine the attitude of the spacecraft. Furthermore, the *control algorithms* produce a command input, so *actuators* are needed to exert the necessary command torques.

### 4.5.1 Sensors

Sensors can be classified in two categories:

- **Absolute attitude sensors**: by observing some celestial body like the Sun, the Earth and the stars, they determine the orientation of the spacecraft with respect to that body. Some examples are horizon sensors, star trackers and magnetometers. In Figure 4.4 the HYDRA-M star tracker by Sodern is shown.
- **Relative attitude sensors**: they are used to measure the angular speed and, by integration, they can be used to calculate the orientation. However, this leads to larger errors, so an absolute sensor is needed for higher accuracy. An example of these sensors are the gyroscopes.

Figure 4.4. Sodern's HYDRA-M star tracker (Author: Christian Lafont, CC-BY-SA)



### 4.5.2 Actuators

The actuators are needed not only to actuate the command input, but also to contrast the perturbing torques acting on the spacecraft (for instance the magnetic field, the solar radiation etc.). The actuators can be divided into the following classes:

- **momentum exchange**: for this purpose, thrusters are the most common example, together with reaction wheels and control moment gyros (CMG);
- **environmental**: solar sails, for instance, produce thrust as a reaction force induced by incident light; another example is the gravity gradient stabilization, that exploits the fact that when an axis of the spacecraft is much longer than the other two then the spacecraft spontaneously orients itself so that the long axis points towards the planet center of mass;
- **dissipative**: they are used to reduce the nutation and the torque disturbances.

It is important to know that each actuator type has a torque range (expressed in $Nm$). Reaction wheels, for instance, can exert a torque from $0.1Nm$ to $1Nm$.

### 4.5.3 Control approaches

Attitude control can be *passive* (when it uses environmental forces such as the gravity gradient), *semi-active* (for instance when reaction wheels or the magnetic field of the Earth is used) and *active* (when thrusters are used). The case considered in the next chapter is a form of active control where thrusters are used as

actuators for the command input. Active control is useful for maneuvering and stabilizing the attitude with precision and speed, and it is also necessary when dealing with external perturbances with non-zero mean.

## 4.6 State equations

So far, two different dynamics were described: the attitude *kinematics*, that express a relationship from $\boldsymbol{\omega}$ and $\mathfrak{q}$, i.e.

$$\dot{\mathfrak{q}} = \frac{1}{2}\mathbf{Q}\boldsymbol{\omega}.$$

and the attitude *dynamics*, that express a relationship from $\mathbf{M}$ to $\boldsymbol{\omega}$, i.e.

$$\dot{\boldsymbol{\omega}} = \mathbf{J}^{-1}\mathbf{M} - \mathbf{J}^{-1}\boldsymbol{\omega} \times \mathbf{J}\boldsymbol{\omega}.$$

Overall, if we imagine to connect in series the two resulting nonlinear systems, we obtain a nonlinear system where $\mathbf{M}$ is the input and $\mathfrak{q}$ is the output. In such a system, the following variables are of interest:

- $\mathbf{w} = (w_1, w_2, w_3)$
- $\mathfrak{q} = (q_0, \mathbf{q}) = (q_0, q_1, q_2, q_3)$
- $\mathbf{x} = (\mathbf{w}, \mathfrak{q})$
- $\mathbf{u} = (u_1, u_2, u_3)$

The purpose of attitude control is making the attitude quaternion $\mathfrak{q}$ converge to a reference quaternion $\mathfrak{q}_r$, and the angular velocity $\mathbf{w}$ converge to $\mathbf{w}_r$.

The state equations of the system that we wish to control are the following:

$$\dot{\mathfrak{q}} = \frac{1}{2}\mathbf{Q}\mathbf{w} \tag{4.27}$$

$$\dot{w} = -\mathbf{J}^{-1}\mathbf{w} \times \mathbf{J}\mathbf{w} + \mathbf{J}^{-1}\mathbf{u} \tag{4.28}$$

where

$$\mathbf{Q} := \begin{bmatrix} -q_1 & -q_2 & -q_3 \\ q_0 & -q_3 & q_2 \\ q_3 & q_0 & -q_1 \\ -q_2 & q_1 & q_0 \end{bmatrix}, \quad \mathbf{w}\times := \begin{bmatrix} 0 & -w_3 & w_2 \\ w_3 & 0 & -w_1 \\ -w_2 & w_1 & 0 \end{bmatrix}$$

We are interested in particular in attitude regulation, i.e. when

$$\mathfrak{q}_r = \text{const and } \mathbf{w}_r = 0.$$

The angular velocity tracking error is defined as

$$\tilde{\mathbf{w}} := \mathbf{w}_r - \mathbf{w},$$

whereas the quaternion tracking error is

$$\tilde{\mathfrak{q}} := \mathfrak{q}^{-1} \otimes \mathfrak{q}_r = \mathfrak{q}^* \otimes \mathfrak{q}_r.$$

We can use a simple linear control law such as the following:

$$\mathbf{u} = k_p \tilde{\mathbf{q}} - k_d \mathbf{w} \tag{4.29}$$

with $k_p > 0, k_d > 0$. Using this law, the state equations of the closed-loop system become

$$\dot{\tilde{\mathfrak{q}}} = -\frac{1}{2} \mathbf{w}^q \otimes \tilde{\mathfrak{q}}, \tag{4.30}$$

$$\dot{\mathbf{w}} = \mathbf{J}^{-1}(-\mathbf{w} \times \mathbf{J}\mathbf{w} + k_p \tilde{\mathbf{q}} - k_d \mathbf{w}). \tag{4.31}$$

There are two equilibrium points, $(\tilde{q_0}, \tilde{\mathbf{q}}, \mathbf{w}) = (\pm 1, \mathbf{0}, \mathbf{0})$, and both signs correspond to the same attitude. The Lyapunov function

$$V = \frac{1}{4} \mathbf{w}^T \mathbf{J} \mathbf{w} + \frac{1}{2} k_p \tilde{\mathbf{q}}^T \tilde{\mathbf{q}} + \frac{1}{2} k_p (1 \mp \tilde{q}_0)^2 \tag{4.32}$$

can be used to prove that, for any initial condition $(\tilde{q}_0(0), \tilde{\mathbf{q}}(0), \mathbf{w}(0))$,

$$\lim_{t \to \infty} (\tilde{q}_0(t), \tilde{\mathbf{q}}(t), \mathbf{w}(t)) = (\pm 1, \mathbf{0}, \mathbf{0}).$$

# Chapter 5

# Design of a DVS for attitude control

## 5.1 Case specifications

The considered case consists of a spacecraft on an Earth orbit, with inertia tensor

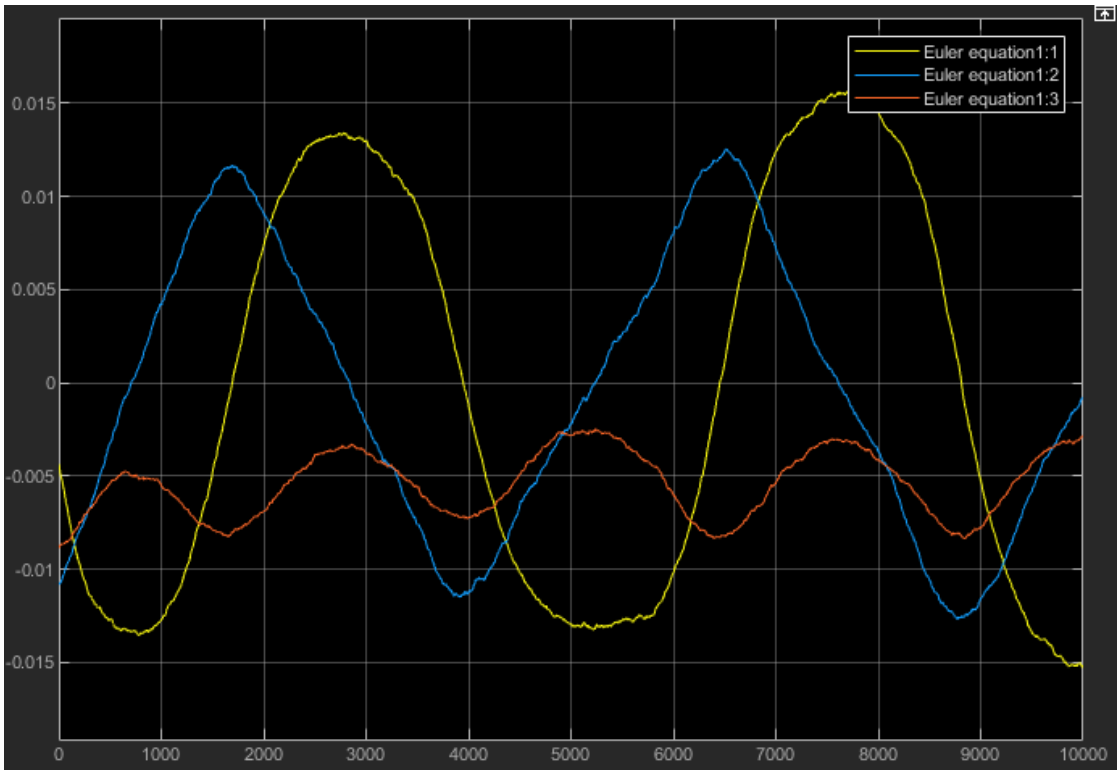
$$\mathbf{J} = \begin{bmatrix} 10000 & 0 & 0 \\ 0 & 9000 & 0 \\ 0 & 0 & 12000 \end{bmatrix} kg\ m^2. \tag{5.1}$$

Two disturbances, $\mathbf{d}^u \sim \mathcal{N}_3(0, \sigma^2)$ and $\mathbf{d}^y \sim \mathcal{N}_4(0, \sigma^2)$ act on the input moment and on the system output respectively, with $\sigma = 0.0001$. The angular velocity $\boldsymbol{\omega}$ is supposed not to be measurable, so the system output only consists of $\mathfrak{q}$. Non-inertial effects, gravity gradient moment, third body gravity, atmosphere drag and solar radiation are neglected. The three thrusters are assumed to be able to give a maximum input moment magnitude $|M_{i,max}| = 1$. The task to be accomplished is to be able to track any reference quaternion $\mathfrak{q}_r$ by means of a controller that takes as input both $\mathfrak{q}$ (measured) and $\boldsymbol{\omega}$ (unmeasurable), so a state observer needs to be deployed. The common way to solve this problem is by using an Extended Kalman Filter (see Appendix), which is a nonlinear extension of the Kalman Filter. The variance of the estimation error obtained with the EKF is compared to the one of a DVS. It is supposed that, when hooking debris, the inertia tensor can reach $3\mathbf{J}$, so the observer must be robust for any matrix $k\mathbf{J}, k \in [1,3]$. Finally, the controller is obtained by choosing $k_p = 10$ and $k_d = 200$.

## 5.2 Training phase

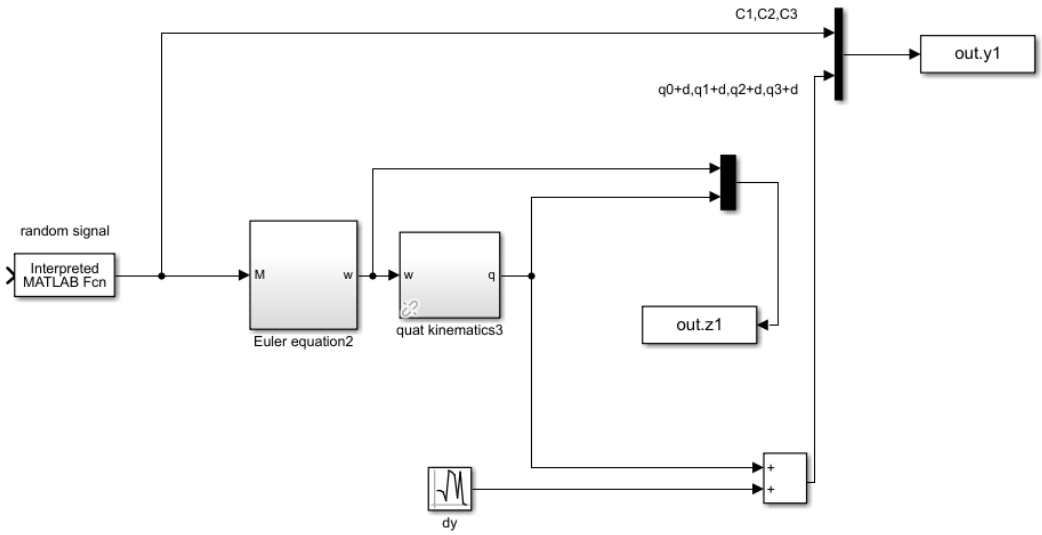
The training set is built performing 50 simulations on the open-loop system. Of these 50 simulations, 25 are done using $\mathbf{J}$ as the inertia tensor, and 25 are done using $3\mathbf{J}$. The simulations are done for $10000s$ (long enough to capture the dynamics of the system, as shown from the angular speed in the open-loop simulation in Figure 5.1) using the Simulink scheme in Figure 5.2 and then are merged. A random input $\mathbf{u} \sim U_3(-1,1)$ is used to stimulate the dynamics of the system (the interval $(-1,1)$ is chosen since the thrusters saturate at $1Nm$).

Figure 5.1. Angular speed behaviour in open-loop with uniform random input




The DVS is designed this time using not the System Identification Toolbox but the Deep Learning Toolbox, as the training algorithms are faster and more adapt to the dimension of the problem, but equivalent in the results. The neural-network used is a *Time-Delay Neural Network (TDNN)*. The TDNN follows the same principle of the neural networks used so far, i.e. it models the dynamic behaviour by using delayed values of the input. The TDNN designed for the DVS is shown in Figure 5.3. The hidden layer uses the **tansig**, i.e. the **Hyperbolic**

Figure 5.2. Open loop Simulink scheme used for training

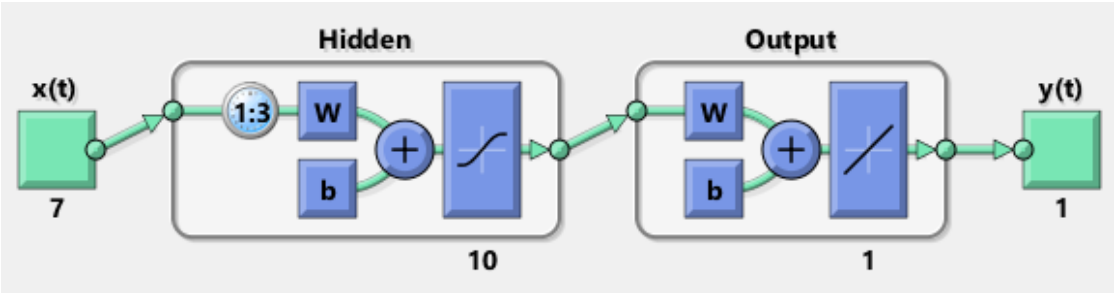



**tangent sigmoid**, as activation function. The *tansig* is defined as

$$tansig(x) = \frac{2}{1 + e^{-2x}} - 1 \tag{5.2}$$

and is very similar to the *sigmoid* function. The second layer uses instead a linear activation to produce the output. The structure is therefore equivalent to the sigmoid network in the System Identification Toolbox.

Figure 5.3. Time Delay Neural Network used for the DVS




To optimize the accuracy, a DVS for each $\omega_i$ was built, instead of a single DVS for $\boldsymbol{\omega}$. This design choice is due to the fact that, in the latter case, the solver would try to train the neural network to find a tradeoff for the accuracy of the

three outputs, whereas with 3 distinct neural networks we can optimize each one at predicting a single output.

The DVS is designed considering:

- the input moment $\mathbf{M}$ and the corrupted measured quaternion $\mathfrak{q}$ as the filter input $\mathbf{u} = (\mathbf{M}, \mathfrak{q}) = (u_1, u_2, u_3, u_4, u_5, u_6, u_7)$
- the real $\boldsymbol{\omega} = (\omega_1, \omega_2, \omega_3)$ as the filter output $\mathbf{y} = (y_1, y_2, y_3)$
- a TDNN with 10 sigmoid units
- a sampling time $T_s = 1s$
- a regularization factor $\lambda = 1e - 8$
- the Levenberg-Marquardt algorithm as training function
- the values $u_1(t-\tau), u_2(t-\tau), \ldots, u_7(t-\tau), \quad \tau = 1, \ldots, 3$ as regressors (the absence of the current values is due to the nature of the state equations, where it is evident that the current values of $\mathfrak{q}$ and $M$ are not important to the current values of $\boldsymbol{\omega}$.
- the first 8000 samples of each experiment as training set, and the last 2000 samples as validation set
- $\boldsymbol{\omega}_0 \sim \mathcal{N}_3(0,0.0001)$ and $\mathfrak{q}_0 \sim \mathcal{N}_4(0,1)$ ($\mathfrak{q}_0$ is then normalized).

Using a validation set in the training allows the optimizer to automatically avoid overfitting, by stopping the training when the training accuracy either exceeds the validation accuracy by far or stops improving, as in Figure 5.4.

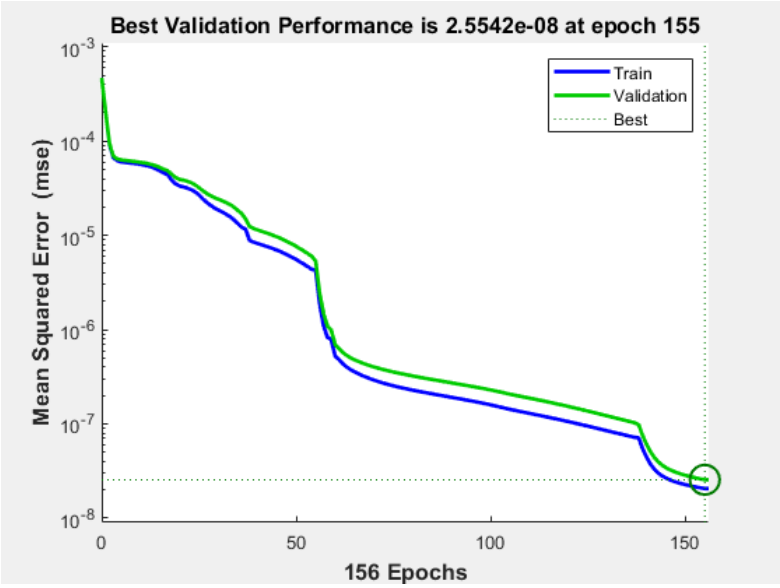
The plots in Figure 5.2 show as an example the nonlinearity of the output of the first filter varying $M_1(t-1)$ and $M_2(t-1)$ (left) or $q_0(t-1)$ and $q_1(t-1)$ (right). The plot on the left is a plane, that is coherent with the linear dependency from $\boldsymbol{w}$ and $\mathbf{M}$ in the state equations; the plot on the right is instead evidently nonlinear, which is coherent too with the state equations.

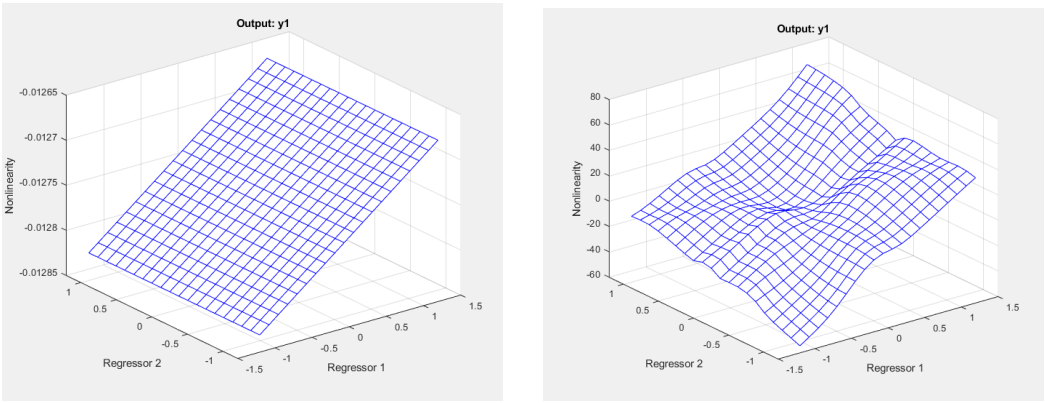
The three neural networks obtained with this procedure constitute together a filter that will be referred to as **T-DVS** (TDNN DVS).

A second architecture is then proposed, where the DVS is designed using a more complex neural network, i.e. using two hidden nonlinear layers instead of just one. Using a TDNN also in this case, 10 sigmoid units are chosen in the first layer and 5 in the second layer. The hyperparameters are the same of the T-DVS. The architecture is shown in Figure 5.6. The filter obtained with this structure will be referred to as **DT-DVS** (Deep TDNN DVS).

Finally, the last architecture proposed uses a TDNN with just one layer, like the TDNN DVS, but here also the current values of $u_1, \ldots, u_7$ are used. This choice

Figure 5.4. Training and validation error during training for the first DVS



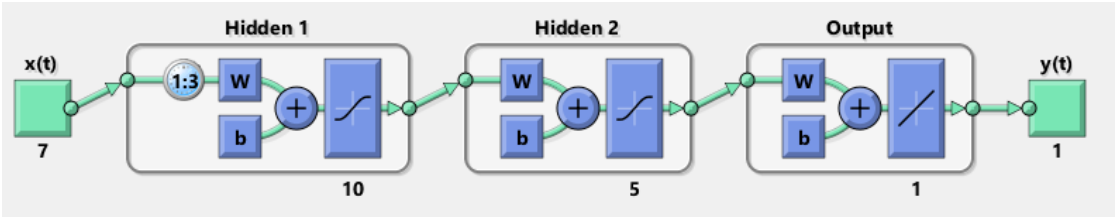

(a) $y_1(M_1(t-1), M_2(t-1))$ (b) $y_1(q_0(t-1), q_1(t-1))$

Figure 5.5. Plot of the nonlinearity of the DVS (Spacecraft)

requires however to use more sigmoid units; 15 units were therefore chosen. The filter obtained with this structure will be referred to as **CT-DVS** (Current TDNN DVS). A remark has to be done about the fact that, in Simulink, an algebraic loop occurs when the DVS estimate is fed back to the controller. When using a fixed-step simulation in Simulink, this does not constitute a problem when the DVS does not use current values; using the CT-DVS, instead, the simulation can

fail (since what is happening is that the DVS is using current values to produce outputs that are fed back in the loop in the same time instant). This problem was solved by putting a unit delay ($z^{-1}$) on the DVS output, so that the algebraic loop is broken. This is taken into account when measuring the variance of the estimation error, that for the CT-DVS is defined as $y_{DVS_i}(t-1) - \omega_i(t)$, whereas for the T-DVS and the DT-DVS it is defined as $y_{DVS_i}(t) - \omega_i(t)$.

Figure 5.6. Deep Time Delay Neural Network used for the DVS




## 5.3 Closed-loop and open-loop performance

The variances of the estimation error of the three DVS architectures and the extended Kalman filter are compared, first with $k = 1$ and then with $k = 3$. 50 experiments are done for 100000 seconds. The EKF takes as input the command $\mathbf{M}$ and the corrupted measurement of the quaternion, and estimates the state $(\boldsymbol{\omega}, \mathbf{q})$. However, only the estimate of $\boldsymbol{\omega}$ is used, and the real quaternion is used for the controller instead of the estimated one, so the quaternion estimate is ignored; this is done to make the comparison fair and equal, since the DVS only estimates $\boldsymbol{\omega}$. First, the performance in closed-loop (i.e. on the controlled system) is calculated. The average variance over 50 experiments is shown in Table 5.1.

| k | $\omega_i$ | T-DVS | DT-DVS | CT-DVS | EKF |
|---|---|---|---|---|---|
| 1 | $\omega_1$ | 1.6e-6 | **5.2e-7** | 7.1e-7 | 2.7e-4 |
| 1 | $\omega_2$ | 1.7e-6 | **4.3e-7** | 5.8e-7 | 1.2e-3 |
| 1 | $\omega_3$ | 1.5e-6 | **6.1e-7** | 6.9e-7 | 2.7e-3 |
| 3 | $\omega_1$ | 8.6e-6 | **5.1e-6** | 6.5e-6 | 3.3e-4 |
| 3 | $\omega_2$ | 9.3e-6 | **4.8e-6** | 6.7e-6 | 7.1e-4 |
| 3 | $\omega_3$ | 7.1e-6 | **5.5e-6** | 6.1e-6 | 1.4e-3 |

Table 5.1. Average error variance of DVS and EKF in closed-loop

Then, the performance in open-loop (without control) is calculated. The results

are shown in Table 5.2.

| k | $\omega_i$ | T-DVS | DT-DVS | CT-DVS | EKF |
|---|---|---|---|---|---|
| 1 | $\omega_1$ | 8.2e-5 | **7e-5** | 8.7e-5 | 2.2e-4 |
| 1 | $\omega_2$ | 5.2e-5 | **4.1e-5** | 5.3e-5 | 1.7e-4 |
| 1 | $\omega_3$ | 2.2e-5 | **1.2e-5** | 2.4e-5 | 1.6-4 |
| 3 | $\omega_1$ | 2e-5 | **1.8e-5** | 2.2e-5 | 7.2e-5 |
| 3 | $\omega_2$ | 1e-5 | **9.9e-6** | 1.4e-5 | 5.5e-5 |
| 3 | $\omega_3$ | 2.8e-6 | **2.7e-6** | 3.9e-6 | 2.9e-5 |

Table 5.2. Average error variance of DVS and EKF in open-loop

All the three DVSs perform better than the EKF, both when $k = 1$ and $k = 3$, which is surprising. Furthermore, the DT-DVS is always the best, both in open-loop and in closed-loop. However, it is evident that the performance is not much higher than the one of the T-DVS, that is less complex. It is also interesting to note that, for almost any initial conditions, using the EKF in closed-loop does not make the quaternion ever converge to the reference; using a DVS, instead, guarantees the convergence. In Figure 5.7, for instance, the satellite attitude in closed-loop when $k = 3$ is shown when using a T-DVS and an EKF, with a quaternion reference $\mathfrak{q}_\mathfrak{r} = [-0.18, 0.16, -0.53, -0.99]$ and random initial conditions.

In Figure 5.8, finally, the estimation error for the three angular speeds with $k = 3$ when using a T-DVS and an EKF is shown.

Figure 5.7. Spacecraft attitude in closed-loop with $k = 3$ using a T-DVS (top) and an EKF (bottom)

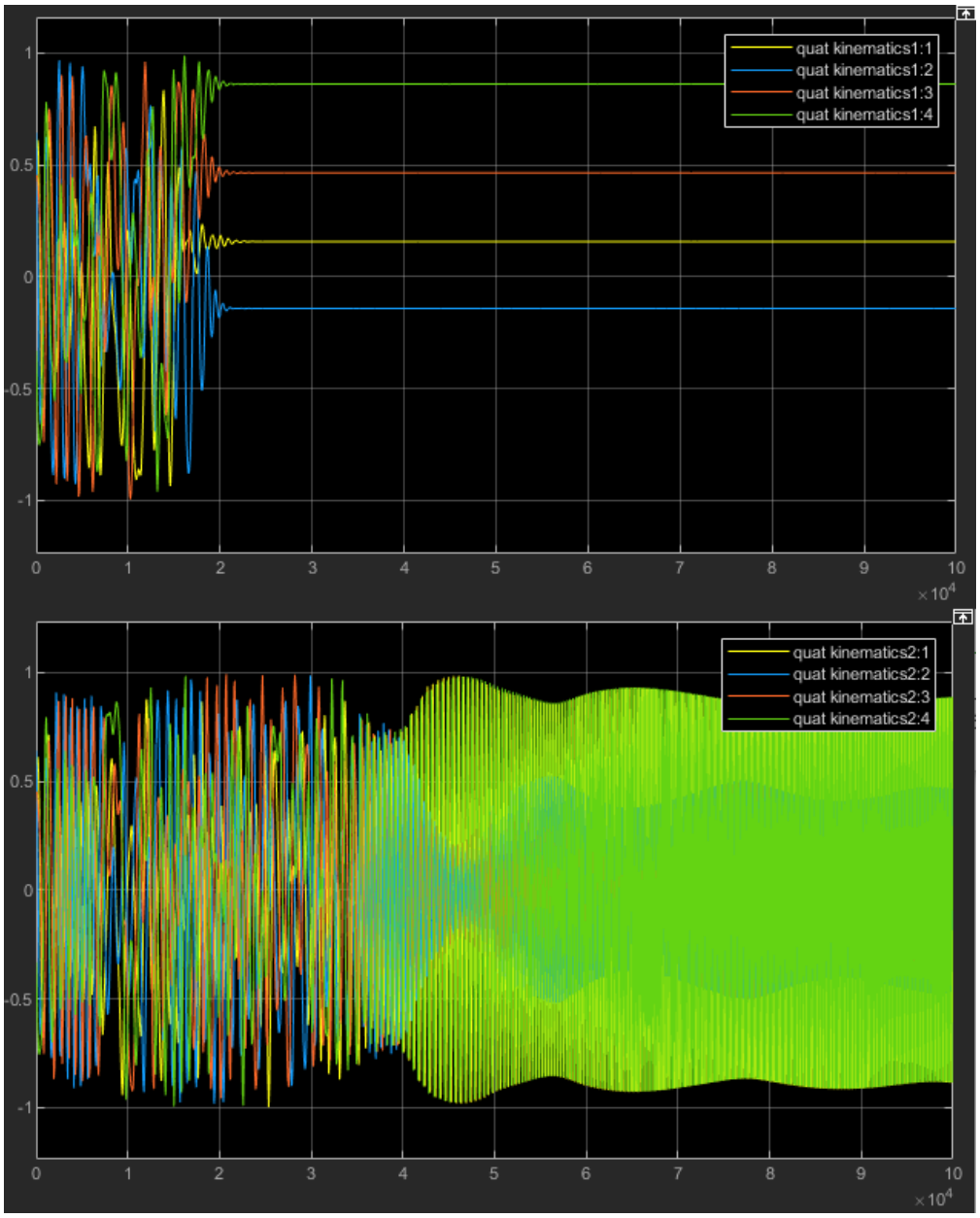
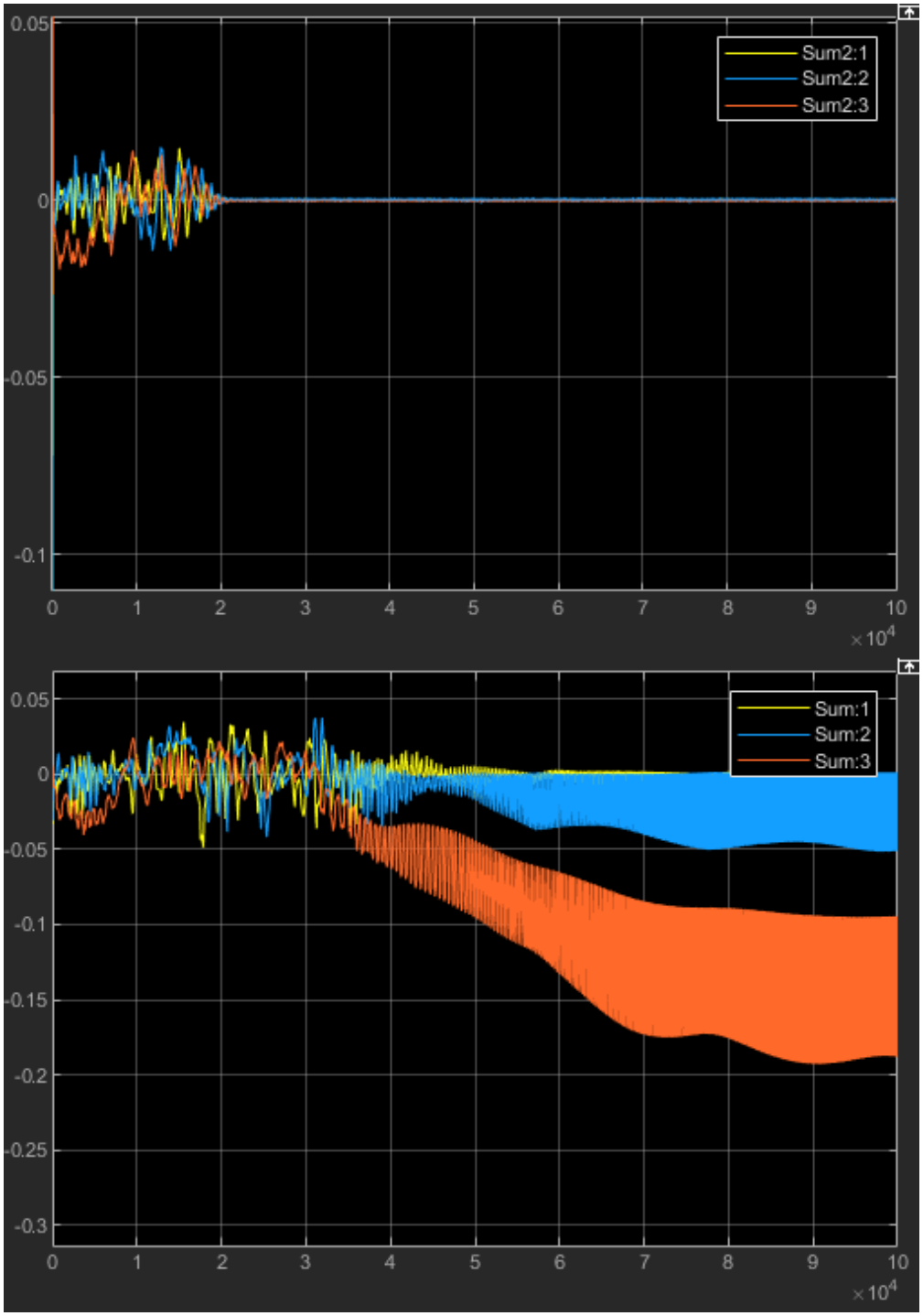


Figure 5.8. Estimation error in closed-loop with $k = 3$ using a T-DVS (top) and an EKF (bottom)



# Chapter 6

# Conclusions

The experiments performed on the two examples of linear systems proved that an observer designed with the one-step procedure and using a neural network can outperform other robust filters found in the literature, when the system is different than the nominal one; furthermore, the use of a neural network proves that the observer can be nonlinear even when the system to filter is linear. In the case of nonlinear systems, the experiments performed on the spacecraft example show with strong evidence than the one-step approach is better than the state of the art used in the industry, since it outperforms the EKF even on the nominal system and does not fail where the EKF fails instead. This presents a double advantage: not only the DVS is easier to implement and does not require to have any prior knowledge on the system to filter or on the uncertainty structure, but it is even better than more complicated filters that are used nowadays in the industry. The method explored in this work, therefore, has consequences of relevant importance for the domain of robust filtering, and it can be used instead of all the other approaches because, even though they are still the standard, the one proposed in this thesis is evidently superior, more robust and better performing.

# Appendix A

# Linear Kalman Filter

It is known that, for linear systems, linear Kalman filters obtain the best performances (i.e. minimum error variance) on the nominal system. However, when the system is uncertain, the performance may decrease substantially.

In discrete-time, we can build two different types of linear Kalman filters: the current one and the delayed one. In both cases, we have that, when the plant is in this form:

$$\begin{aligned}\mathbf{x}_{k+1} &= \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k + \mathbf{G}\mathbf{w}_k \\ \mathbf{y}_k &= \mathbf{C}\mathbf{x}_k + \mathbf{D}\mathbf{u}_k + \mathbf{H}\mathbf{w}_k + \mathbf{v}_k\end{aligned}$$

where $\mathbf{w}$ and $\mathbf{v}$ are noises that satisfy:

$$\begin{aligned}\mathbb{E}(\mathbf{w}_k\mathbf{w}_k^T) &= \mathbf{Q} \\ \mathbb{E}(\mathbf{v}_k\mathbf{v}_k^T) &= \mathbf{R} \\ \mathbb{E}(\mathbf{w}_k\mathbf{v}_k^T) &= \mathbf{N}\end{aligned}$$

then the discrete-time Kalman filter has this state equation:

$$\hat{\mathbf{x}}_{k+1|k} = \mathbf{A}\hat{\mathbf{x}}_{k|k-1} + \mathbf{B}\mathbf{u}_k + \mathbf{L}(\mathbf{y}_k - \mathbf{C}\hat{\mathbf{x}}_{k|k-1} - \mathbf{D}\mathbf{u}_k)$$

where the gain matrix L is the solution of the discrete Riccati equation:

$$\mathbf{L} = (\mathbf{A}\mathbf{P}\mathbf{C}^T + \overline{\mathbf{N}})(\mathbf{C}\mathbf{P}\mathbf{C}^T + \overline{\mathbf{R}})^{-1}$$

with

$$\begin{aligned}\overline{\mathbf{R}} &= \mathbf{R} + \mathbf{H}\mathbf{N} + \mathbf{N}^T\mathbf{H}^T + \mathbf{H}\mathbf{Q}\mathbf{H}^T \\ \overline{\mathbf{N}} &= \mathbf{G}(\mathbf{Q}\mathbf{H}^T + \mathbf{N})\end{aligned}$$

Now, in the case of delayed filtering, we just have that

$$\begin{bmatrix}\hat{\mathbf{x}}_{k|k-1} \\ \hat{\mathbf{y}}_{k|k-1}\end{bmatrix} = \begin{bmatrix}\mathbf{I} \\ \mathbf{C}\end{bmatrix} \hat{\mathbf{x}}_k|k-1 + \begin{bmatrix}\mathbf{0} & \mathbf{0} \\ \mathbf{D} & \mathbf{0}\end{bmatrix} \begin{bmatrix}\mathbf{u}_k \\ \mathbf{y}_k\end{bmatrix} \tag{A.1}$$

In the case of current filtering, the state estimate $\hat{\mathbf{x}}_{k|k-1}$ is updated using the new measurement $\mathbf{y}_k$:

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{M}_x(\mathbf{y}_k - \mathbf{C}\hat{\mathbf{x}}_{k|k-1} - \mathbf{D}\mathbf{u}_k)$$

and $\hat{\mathbf{y}}_{k|k-1}$ is updated too:

$$\hat{\mathbf{y}}_{k|k} = \mathbf{C}\hat{\mathbf{x}}_{k|k-1} + \mathbf{D}\mathbf{u}_k + \mathbf{M}_y(\mathbf{y}_k - \mathbf{C}\hat{\mathbf{x}}_{k|k-1} - \mathbf{D}\mathbf{u}_k).$$

$\mathbf{M}_x$ and $\mathbf{M}_y$ are the innovation gains, defined as

$$\begin{aligned}\mathbf{M}_x &= \mathbf{P}\mathbf{C}^T(\mathbf{C}\mathbf{P}\mathbf{C}^T + \overline{\mathbf{R}})^{-1} \\ \mathbf{M}_y &= (\mathbf{C}\mathbf{P}\mathbf{C}^T + \mathbf{H}\mathbf{Q}\mathbf{H}^T + \mathbf{H}\mathbf{N})(\mathbf{C}\mathbf{P}\mathbf{C}^T + \overline{\mathbf{R}})^{-1}.\end{aligned}$$

# Appendix B

# Extended Kalman Filter

It is known that, for linear systems, the linear Kalman filter is the optimal (minimum variance) filter. In the case of nonlinear systems, however, the linear Kalman filter cannot be used since it does not model the nonlinearities and cannot therefore perform an accurate estimation of the states. The **extended Kalman filter** is used instead, which is based on a step-by-step linearization of the system equations along the trajectory. Since in a real-world application a filter is implemented on a digital device, let us consider a discrete-time formulation. If we have a discrete-time nonlinear system in the form

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k) + \mathbf{d}_k, \quad \mathbf{y}_k = h(\mathbf{x}_k) + \mathbf{d}_k^y, \quad k \in \mathbf{N} \tag{B.1}$$

where $\mathbf{x}_k \in \mathbf{R}^{n_x}$ is the state vector, $\mathbf{u}_k \in \mathbf{R}^{n_u}$ is the input, $\mathbf{y}_k \in \mathbf{R}^{n_y}$ is the output, $\mathbf{d}_k \in \mathbf{R}^{n_x}$ is the disturbance and $\mathbf{d}_k^y \in \mathbf{R}^{n_y}$ is the measurement noise. It is supposed that only $\mathbf{y}_k$, $\mathbf{u}_k$ are measurable.
State estimation, as usual, consists in computing a prediction $\mathbf{x}_k^p$ of $\mathbf{x}_k$

$$\mathbf{x}_k^p = f(\hat{\mathbf{x}}_{k-1}, \mathbf{u}_{k-1}), \tag{B.2}$$

and finally improve it using the current output measurement

$$\hat{\mathbf{x}}_k = \mathbf{x}_k^p + \mathbf{K}_k \Delta \mathbf{y}_k, \tag{B.3}$$

with $\Delta \mathbf{y}_k = \mathbf{y}_k - h(\mathbf{x}_k^p)$.
If the system was linear, we could write

$$f(\mathbf{x}_k, \mathbf{u}_k) = \mathbf{F}_k \mathbf{x}_k + \mathbf{G}_k \mathbf{u}_k, \quad h(\mathbf{x}_k) = \mathbf{H}_k \mathbf{x}_k. \tag{B.4}$$

(the linear Kalman filter, for instance, computes $\mathbf{K}_k$ from $\mathbf{F}_k, \mathbf{H}_k$ and in such a way that $\mathbb{E}(\|\mathbf{x}_k - \hat{\mathbf{x}}_k\|_2^2)$ is minimized).

Since the system is nonlinear instead, the matrices are recomputed at each step, linearizing the system equations:

$$\mathbf{F}_k := \mathbf{J}_f(\mathbf{x}_k, \mathbf{u}_k) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_{n_x}} & \frac{\partial f_1}{\partial u_1} & \cdots & \frac{\partial f_1}{\partial u_{n_u}} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \frac{\partial f_{n_x}}{\partial x_{n_x}} & \cdots & \frac{\partial f_{n_x}}{\partial x_{n_x}} & \frac{\partial f_{n_x}}{\partial u_1} & \cdots & \frac{\partial f_{n_x}}{\partial u_{n_u}} \end{bmatrix} (\mathbf{x}_k, \mathbf{u}_k) \quad \in \mathbf{R}^{n_x, n_x+n_u} \tag{B.5}$$

$$\mathbf{H}_k := \mathbf{J}_h(\mathbf{x}_k) = \begin{bmatrix} \frac{\partial h_1}{\partial x_1} & \cdots & \frac{\partial h_1}{\partial h_{n_x}} \\ \cdots & \cdots & \cdots \\ \frac{\partial h_{n_y}}{\partial x_{n_x}} & \cdots & \frac{\partial h_{n_y}}{\partial x_{n_x}} \end{bmatrix} (\mathbf{x}_k) \quad \in \mathbf{R}^{n_y, n_x} \tag{B.6}$$

Defining the following quantities:

- $\mathbf{x}_k^p$: prediction of $\mathbf{x}_k$ computed at step $k-1$
- $\hat{\mathbf{x}}_k$: corrected estimate of $\mathbf{x}_k$ computed at step $k$
- $\mathbf{P}_k := \mathbb{E}((\mathbf{x}_k - \hat{\mathbf{x}}_k)(\mathbf{x}_k - \hat{\mathbf{x}}_k)^T)$: covariance matrix of $\mathbf{x}_k - \hat{\mathbf{x}}_k$
- $\mathbf{Q}^d := \mathbb{E}(\mathbf{d}_k \mathbf{d}_k^T)$: covariance matrix of $\mathbf{d}_k$
- $\mathbf{R}^d := \mathbb{E}(\mathbf{d}_k^y (\mathbf{d}_k^y)^T$: covariance matrix of $\mathbf{d}_k^y$

the first step is choosing $\mathbf{Q}^d$ and $\mathbf{R}^d$. They are typically chosen as diagonal matrices with the variances of $\mathbf{d}_k, \mathbf{d}_k^y$ on the diagonal.
Then, $\hat{\mathbf{x}}_0$ (estimated initial state) and $\mathbf{P}_0$ (estimated initial covariance matrix) are initialied, typically with values $\mathbf{0}$ and $\mathbf{I}$ (identity matrix).
The algorithm consists in the following two steps:

- Prediction:

$$\mathbf{x}_k^p = f(\hat{\mathbf{x}}_{k-1}, \mathbf{u}_{k-1}) \tag{B.7}$$

$$\mathbf{P}_k^p = \mathbf{F}_{k-1} \mathbf{P}_{k-1} \mathbf{F}_{k-1}^T + \mathbf{Q}^d \tag{B.8}$$

- Update:

$$\mathbf{S}_k = \mathbf{H}_k \mathbf{P}_k^p \mathbf{H}_k^T + \mathbf{R}^d \tag{B.9}$$

$$\mathbf{K}_k = \mathbf{P}_k^p \mathbf{H}_k^T \mathbf{S}_k^{-1} \tag{B.10}$$

$$\Delta \mathbf{y}_k = \mathbf{y}_k - h(\mathbf{x}_k^p) \tag{B.11}$$

$$\hat{\mathbf{x}}_k = \mathbf{x}_k^p + \mathbf{K}_k \Delta \mathbf{y}_k \tag{B.12}$$

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \mathbf{P}_k^p. \tag{B.13}$$

In the prediction step, the nonlinear model equations are used to obtain a first prediction of the future step. In the update step, then, the prediction is corrected using the current output measurement. $\mathbf{P}_k$ is recomputed at each step by linearizing the system equations along the trajectory, making $\mathbf{K}_k$ change too.

# Bibliography


[1] Sing Kiong Nguang and Minyue Fu. Robust nonlinear h$\infty$ filtering. *Automatica*, 32(8):1195–1199, August 1996. doi: 10.1016/0005-1098(96)00067-2. URL https://doi.org/10.1016/0005-1098(96)00067-2.

[2] Bor-Sen Chen, Wen-Hao Chen, and Hsuan-Liang Wu. Robust $h_2/h_\infty$ global linearization filter design for nonlinear stochastic systems. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 56(7):1441–1454, July 2009. doi: 10.1109/tcsi.2008.2007059. URL https://doi.org/10.1109/tcsi.2008.2007059.

[3] Y. Theodor and U. Shaked. Robust discrete-time minimum-variance filtering. *IEEE Transactions on Signal Processing*, 44(2):181–189, 1996. doi: 10.1109/78.485915. URL https://doi.org/10.1109/78.485915.

[4] Zhisheng Duan, Jingxin Zhang, Cishen Zhang, and Edoardo Mosca. Robust h2 and hinfinity filtering for uncertain linear systems. *Automatica*, 42:1919 – 1926, 2006. ISSN 0005-1098. doi: 10.1016/j.automatica.2006.06.004.

[5] Carlo Novara, Mario Milanese, Eilyan Bitar, and Kameshwar Poolla. The filter design from data (FD2) problem: parametric-statistical approach. 22 (16):1853–1872, September 2011. doi: 10.1002/rnc.1791. URL https://doi.org/10.1002/rnc.1791.

[6] Lennart Ljung. *System Identification (2nd Ed.): Theory for the User*. Prentice Hall PTR, USA, 1999. ISBN 0136566952.

[7] Carlo Novara. *Nonlinear Control and Aerospace applications: lecture notes. Politecnico di Torino*. 2020.