UNIVERSITÀ DEGLI STUDI DI UDINE

Facoltà di Ingegneria

Corso di Laurea in Ingegneria Elettronica

Dipartimento di Matematica e Informatica

Laboratorio di Dinamica dei Sistemi

Tesi di Laurea

EQUALIZED FILTERING OF DISCRETE–TIME PROCESSES

Filtraggio equalizzato di sistemi discreti

Relatore

Chiar.mo Prof. Franco Blanchini

Laureanda

Giulia Giordano

Anno Accademico 2009/2010

Summary

A filter provides an estimation of an unknown quantity, typically an output of a dynamical system. The filter design procedure aims at minimizing the estimation error with respect to some norm. The present work deals with equalized filtering for discrete–time processes, in presence of unknown – but bounded – noise, an approach introduced by Blanchini and Sznaier [1], which allows for the synthesis of bounded complexity filters. The main idea of their paper is to attain a *fixed order* recursive filter, for systems subject to ℓ^∞ bounded disturbances, such that the estimation error is guaranteed to be asymptotically confined in a hyperrectangle whose "size" should be minimized. As it is proved in [1], the problem can be solved by means of linear, time–invariant filters, which can be synthesized by solving a convex optimization feasibility problem.

The aim of this thesis is to:

- analyse the class of filters proposed in [1];
- apply these filters to systems affected by noise *also in presence of control or, more in general, of a known signal*;
- test these filters on several examples and compare their performance with that achieved by means of standard Kalman filters.

A MATLAB code has been implemented to simulate numeric examples.

The simulations have evidenced a good behaviour: the filter obtained with the procedure explained in [1] typically outperforms the performance of the Kalman filter, which is statistically optimal, if compared in terms of the worst case error.

Contents

List of Figures

Chapter 1

Introduction

Problem Statement and Goal

We consider linear, time–invariant systems subject to ℓ^∞ bounded disturbances. Two types of noises are considered: a process noise, that affects directly the system evolution, and a measurement noise, that affects the measurement output. The goal is to estimate a certain system variable, the estimated output, which is in general different from the measured one. Clearly, the norm of the estimation error, defined as the difference between the estimated output and the estimating variable, has to be rendered as small as possible.

Our purpose is to synthesize filters – of *bounded complexity* – which assure that the error is asymptotically confined within the tightest possible hyperrectangle.

Literature Review

When hard bounds on estimation errors are desired and the only information available on exogenous disturbances is a pointwise bound on a suitable norm, the so called deterministic, unknown–but–bounded approach is usually preferred to the classical stochastic estimation methods. The typical goal is to design an estimator that minimizes the worst case estimation error due to exogenous inputs which is only known to belong to a given set.

Among the pioneering papers devoted to this problem, [9] and [10] analysed the state consistency set, i.e. the set of states compatible with both a–priori assumptions and experimental measurements. It was shown that, in the case of ℓ^2 bounded exogenous disturbances, the set of states consistent with the experimental observations is an ellipsoid whose center and covariance matrix can be recursively obtained via a Kalman–filter–like estimator. Yet the property does not hold for pointwise in time (e.g. ℓ^∞ like) constraints on the disturbance: in this case, even constraining the disturbances to belong to an ellipsoid at each point in time does not lead to easily characterizable consistency sets for the states, although these sets can be conservatively overbounded by an ellipsoid.

Worst case estimation in the presence of ℓ^∞ bounded disturbances was studied in [11], [12], [13] (see also the survey [14]). The main result of these papers shows that pointwise optimal estimators can be obtained as the product of a subset of past measurements and a time varying gain. Both the gain and the set of relevant measurements result from solving a linear programming optimization problem. However, this optimization problem involves *all* past measurements. Thus, the complexity of these estimators grows with time. Moreover, the filter is non–recursive: current estimates are obtained by solving a linear programming problem that involves all available information, rather than by propagating past estimates.

The use of nonlinear recursive filters was proposed in [15]: the idea is to bound the set of possible states consistent with the output observations by a set whose center is propagated recursively and whose shape can be found by solving (at each instant) an optimization problem. Still, the complexity of the resulting observer is potentially high.

A semi–recursive algorithm was proposed in [16]. In the case of known initial conditions, the optimal ℓ^∞ estimation problem is reduced to an ℓ^1 model matching problem ([17], [18], [19]) that can be solved with arbitrary precision by using the techniques in [18]. In the case

of unknown initial conditions, the complete, semi–recursive estimator is obtained by using a non–recursive pointwise optimal estimator similar to that in [13] for the first time steps and switching afterwards to the recursive ℓ^1 estimator. Again, this estimator complexity cannot be bounded a–priori.

An alternative approach involves set–valued observers ([20], [21]): pointwise optimal estimators are obtained by recursively applying the Fourier–Motzkin algorithm to construct a polyhedral set guaranteed to contain the states of the plant. An ℓ^∞ pointwise optimal estimator is then obtained from these sets, by simply using as estimate of the unknown output z the center z_c of the set of all output values compatible with the present set estimate of the state. However, propagation of these estimates is not recursive, e.g. $z_c(k+1)$ cannot be directly constructed from the past estimates $z_c(k-i)$. Moreover, in principle the complexity of the estimator is not bounded a–priori and increases with time.

So, previous work dealing with the problem, based on constructing the consistency set for the states of the plant, led to filters whose complexity can be arbitrarily large and potentially grows online.

Equalized Filtering: Concept and Benefits

Since complexity is very high in observers based upon the idea of propagating a set known to contain the unknown state of the plant, to avoid this difficulty Blanchini and Sznaier ([1]) recently proposed a different approach, based on the idea of *equalized performance* (first introduced in their paper [2], concerning suboptimal ℓ^1 controller design). This new approach allows to easily attain the goal to synthesize *fixed order* recursive filters for systems subject to ℓ^∞ bounded disturbances, with guaranteed worst case estimation error.

Rather than attempting to confine the *state* of the system to a given set, it is more desirable to work directly with the *estimation error*: the new approach leads to the synthesis of *bounded complexity filters linear recursive* that confine the estimation error to the tightest possible hyperrectangle, for a set of suitable initial conditions. For initial conditions outside this set, the estimation error converges, in finite time, to the design value.

As we will see, the simulations performed in this work show a good behaviour of the filters so obtained, also in challenging situations.

Chapters Overview

The first part, dedicated to theoretical preliminaries, includes three chapters: Chapter 2 deals with the new convex optimization approach to synthesizing bounded complexity ℓ^∞ filters, proposed by Blanchini and Sznaier; Chapter 3 extends the application of such filters to systems in presence of control or, more in general, of a known input; Chapter 4 shows how to obtain a discrete–time Kalman filter, in order to compare it with the filter obtained via convex optimization.

In the second part, several examples and applications are presented. The general procedure of filter synthesis and simulation is described in Chapter 5; in Chapter 6, examples of systems affected only by unknown–but–bounded noises are considered, while in Chapter 7 systems with known inputs too are considered. General conclusions follow.

Some source code of the simulation programs is reported in Appendix A and its usage is explained; all the MATLAB code is enclosed in a CD. Appendix B contains some hints about convex optimization derived from [8].

Part I

Theoretical Preliminaries

Chapter 2

Filter Synthesis via Convex Optimization

In this chapter we report results exposed by Blanchini and Sznaier in their paper [1] to synthesize bounded complexity ℓ^∞ filters. Their work provides the theoretical background for the present thesis. For the sake of completeness the essential results from [1] are quoted.

2.1 Notation, Definitions and Preliminary Results

In the thesis we will adopt the following notations.

$\|y\|_\infty$	∞ norm of the vector $y \in R^n$: $\|y\|_\infty \doteq \max_i \|y\|_i$.
$M(j,:)$	j^{th} row of the matrix M.
$\|M\|_1$	$\infty \to \infty$ induced norm of matrix $M \in R^{n\times m}$: $\|M\|_1 \doteq max_i \sum_j \|M_{ij}\|$
ℓ^1_n, ℓ^∞_n	extended Banach spaces of vector valued real sequences $\{y\}_0^\infty \in \mathbb{R}^n$ equipped with the norms $\|y\|_{\ell^1} \doteq \sum_{i=0}^\infty \|y_i\|_\infty$ and $\|y\|_{\ell^\infty} \doteq \sup_i \|y_i\|_\infty$, respectively.
$\mathcal{B}\ell^1$, $\mathcal{B}\ell^\infty$	unit balls in ℓ^1, ℓ^∞.
$\mathcal{B}\ell^\infty(\mu)$	scaled unit ball in ℓ^∞_n. Given $\mu \doteq [\mu_1 \dots \mu_n]$ $$\mathcal{B}\ell^\infty(\mu) \doteq \{e \in \ell^\infty_n : e_i/\mu_i \in \mathcal{B}\ell^\infty\}$$
$\|G\|_{\ell^\infty\to\ell^\infty}$	ℓ^∞ to ℓ^∞ induced norm of the operator $G : \ell^\infty \to \ell^\infty$, e.g. $\|G\|_{\ell^\infty\to\ell^\infty} \doteq \sup_{y\neq 0} \frac{\|Gy\|_{\ell^\infty}}{\|y\|_{\ell^\infty}}$
$Y(\lambda)$	λ–transform of a sequence $\{y_k\}_0^\infty$: $Y(\lambda) \doteq \sum_{k=0}^\infty y_k\lambda^k$

We consider a scalar ARMA (*Auto Regressive Mobile Average*) model of the form

$$y(k) = -\sum_{i=1}^{n} a_i y(k-i) + \sum_{i=0}^{m} b_i v(k-i);\ n \geq m \tag{2.1}$$

in which the value of the output y at time–instant k is computed as a mobile average on a finite set of values of input and output at time–instants before k and of input at time–instant k.

An ARMA model is associated to its λ–transform representation, which corresponds to setting $\lambda = 1/z$ in the usual z–transform representation.

$$y(\lambda) = \frac{\sum_{i=0}^m b_i\lambda^i}{\sum_{i=0}^n a_i\lambda^i} v(\lambda) \tag{2.2}$$

The notion of equalized performance introduced in [2] (see also [3]) is here reported.

Definition 1. *Consider an LTI (*Linear Time Invariant*) plant described by a model of the form* (2.1)*. Given $r \geq n$, the plant is said to achieve an equalized r–performance level μ if, whenever the input and output sequences $\{v\}, \{y\}$ satisfy $|v(t)| \leq 1$ and $|y(t)| \leq \mu$ for all $t = k, k-1, \ldots, k-r+1$, then $\|y(k+1)\| \leq \mu$ (thus $\|y(k+i)\| \leq \mu$, for $i > 0$). In particular the case $r = n$ is simply referred to as equalized performance.*

Any stable plant achieves finite r–equalized performance for some large enough r. Further, if a SISO (*Single Input Single Output*) plant achieves r–performance μ for some finite r, then it achieves r'–performance μ for any $r' > r$.

Some properties concerning the relationship between equalized performance and the ℓ^∞ induced norm are exposed in a lemma that we quote from [3].

Lemma 1. *Given a stable, LTI SISO plant $y(\lambda) = G(\lambda)v(\lambda)$, as in (2.2), with finite r–equalized performance level $\mu(r_o)$ for some $r_o \geq n$, the following holds:*

1. $\|G\|_{\ell^\infty \to \ell^\infty} \leq \mu(r_o)$*, with the equality holding for* Finite Impulse Response *(FIR) plants.*

2. $\mu(r) \downarrow \|G\|_{\ell^\infty \to \ell^\infty}$.

The purpose is

- to design a filter such that, if at some time instant t_o the past r values of the estimation error are contained in a μ–hyperrectangle, then this property will hold for all $t > t_o$ and all disturbances $||v||_{\ell^\infty} \leq 1, \|w\|_{\ell^\infty} \leq 1$

- to obtain the tightest hyperrectangle satisfying this property.

The concept is illustrated in Fig. 2.1.

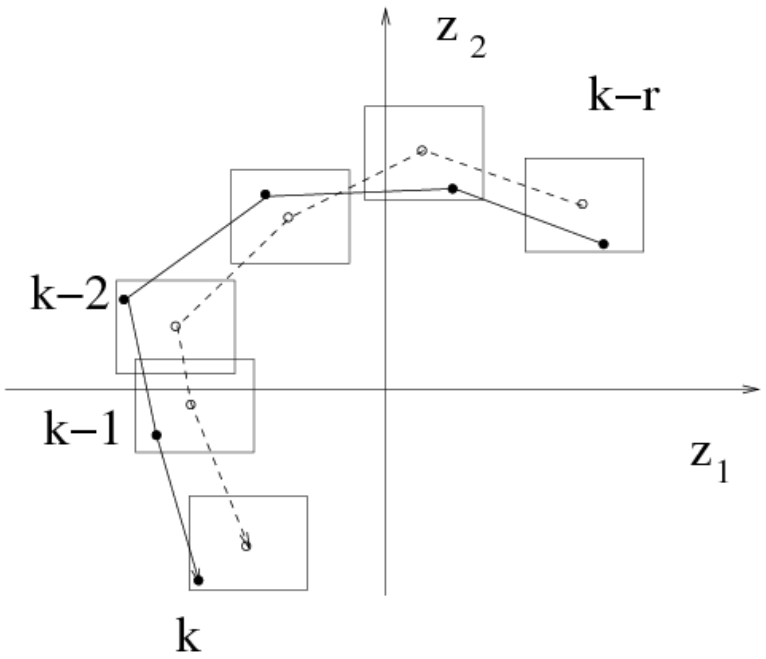


Figure 2.1: The equalized filtering idea: full dots are the true trajectory, empty dots are the estimated trajectory (from [1])

2.2 Problem Statement and Formulation

Consider an LTI plant subject to ℓ^∞ bounded disturbances, with state space realization

$$\begin{aligned} x(k+1) &= Ax(k) + B_v v(k) \\ z(k) &= Hx(k) \\ y(k) &= C_y x(k) + Dw(k) \end{aligned} \tag{2.3}$$

or with λ–transform representation

$$z(\lambda) = \frac{M(\lambda)}{d(\lambda)} v(\lambda) \tag{2.4}$$

$$y(\lambda) = \frac{N(\lambda)}{d(\lambda)} v(\lambda) + Dw(\lambda) \tag{2.5}$$

where $z \in \mathbb{R}^s$ is the output to be estimated, $y \in \mathbb{R}^q$ are the measurements available to the filter, $v \in \mathbb{R}^p$ is process noise and $w \in \mathbb{R}^q$ is measurement noise; besides $d(\lambda) = det(I - \lambda A)$.

As pointed out in [1], the assumption that the plant is strictly proper, with respect to the input v, is made for notational simplicity and it can be removed at the price of a more involved notation in the subsequent development. We will also assume now that z is a scalar, but this assumption will be relaxed later, in section 2.5.

The filter to be designed is of the form

$$\hat{z}(\lambda) = \frac{B(\lambda)}{a(\lambda)} y(\lambda) \tag{2.6}$$

such that the estimation error

$$e(\lambda) = z(\lambda) - \hat{z}(\lambda) \tag{2.7}$$

is confined to a hyperrectangle. The complete filtering scheme is illustrated in Fig. 2.2.

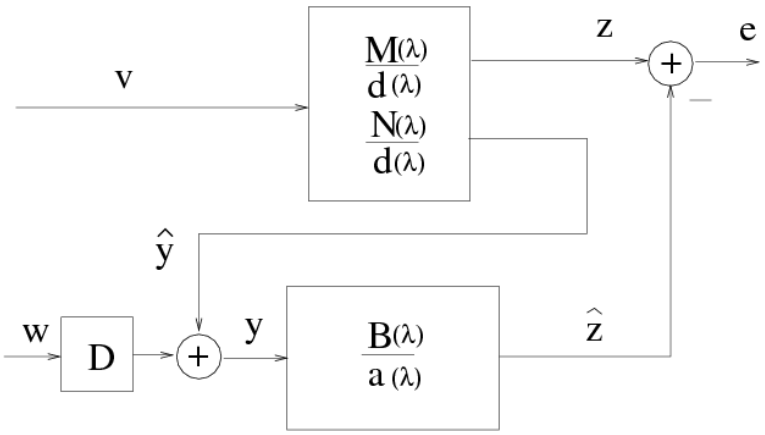


Figure 2.2: The filtering scheme (from [1])

We will consider only filters that belong to the class of generalized Luenberger observers defined as follows [4].

Definition 2. *A system of the form*

$$\xi(k+1) = P\xi(k) + Ly(k) \tag{2.8}$$

$$\hat{x}(k) = Q\xi(k) + Ry(k) \tag{2.9}$$

$$\hat{z}(k) = H\hat{x}(k) \tag{2.10}$$

is a generalized Luenberger (state) observer for system (2.3) *if P is a stable matrix and $\hat{x}(k) - x(k) \to 0$ as $k \to \infty$, when $w(k) \equiv 0$ and $v(k) \equiv 0$.*

Lemma 2. *The system (2.8)–(2.10) is a generalized observer for* (2.3) *if and only if P is stable and there exists a full column rank matrix T such that*

$$\begin{aligned} TA - LC_y &= PT && (2.11)\\ QT + RC_y &= I && (2.12) \end{aligned}$$

Proof. See [4], [5]. □

The standard n–order Luenberger observer corresponds to the choice $T = I$ and $R = 0$. Selecting a "tall" T matrix leads to a higher order observer, with additional degrees of freedom that can be used to optimize performance.

It is easy to show that, if P is stable and (2.11)–(2.12) are valid, $\hat{x}(k) - x(k) \to 0$ as $k \to \infty$, when $w(k) \equiv 0$ and $v(k) \equiv 0$. In fact

$$\begin{aligned} [Tx - \xi](k+1) &= TAx(k) + TB_v v(k) - P\xi(k) - Ly(k)\\ &= TAx(k) + TB_v v(k) - P\xi(k) - LC_y x(k) - LDw(k)\\ &= PTx(k) + TB_v v(k) - P\xi(k) - LDw(k)\\ &= P[Tx - \xi](k) + TB_v v(k) - LDw(k) \end{aligned}$$

Since P is stable, when $w(k) \equiv 0$ and $v(k) \equiv 0$, $[Tx - \xi](k) \to 0$ as $k \to \infty$. Thus

$$\begin{aligned} \hat{x}(k) - x(k) &= Q\xi(k) + R[C_y x(k) + Dw(k)] - Ix(k)\\ &= Q\xi(k) + [RC_y - I]x(k) + RDw(k)\\ &= Q\xi(k) - QTx(k) + RDw(k)\\ &= -Q[Tx - \xi](k) + RDw(k) \end{aligned}$$

So, when $w(k), v(k) \equiv 0$, as $k \to \infty$ $\hat{x}(k) - x(k) \to 0$, because $[Tx - \xi](k) \to 0$.

Restricting the filter to be a generalized observer imposes a constraint on its structure.

Lemma 3. *If the filter* (2.6) *is a generalized state observer for system* (2.3)*, then the polynomial matrices $M(\lambda)$ (of dimension $1 \times p$), $N(\lambda)$ (of dimension $q \times p$), $B(\lambda)$ (of dimension $1 \times q$) and the polynomials $a(\lambda)$ and $d(\lambda)$ satisfy the following condition:*

$$M(\lambda)a(\lambda) - B(\lambda)N(\lambda) = C(\lambda)d(\lambda) \tag{2.13}$$

for some polynomial matrix $C(\lambda)$.

We report, for completeness, the proof written by Blanchini and Sznaier in [1], since some details will be required later.

Proof. From equations (2.8)–(2.12) it follows that

$$\begin{aligned} [Tx - \xi](k+1) &= TAx(k) - P\xi(k) - L[C_y x(k) + Dw(k)] + TB_v v(k)\\ &= P[Tx - \xi](k) + TB_v v(k) - LDw(k)\\ x(k) - \hat{x}(k) &= x(k) - Q\xi(k) - RC_y x(k) - RDw(k)\\ &= Q[Tx - \xi](k) - RDw(k) \end{aligned} \tag{2.14}$$

Furthermore,

$$\begin{aligned} e(k) &= z(k) - \hat{z}(k) = Hx(k) - H\hat{x}(k)\\ &= H[x(k) - \hat{x}(k)] = HQ[Tx - \xi](k) - HRDw(k) \end{aligned}$$

With the change of variables $\eta = x$ and $\vartheta = [Tx - \xi]$, the state space representation of the combined plant–filter system becomes

$$\begin{aligned} \begin{bmatrix} \eta(k+1) \\ \vartheta(k+1) \end{bmatrix} &= \begin{bmatrix} A & 0 \\ 0 & P \end{bmatrix} \begin{bmatrix} \eta(k) \\ \vartheta(k) \end{bmatrix} + \begin{bmatrix} B_v & 0 \\ TB_v & -LD \end{bmatrix} \begin{bmatrix} v(k) \\ w(k) \end{bmatrix}\\ e(k) &= \begin{bmatrix} 0 & HQ \end{bmatrix} \begin{bmatrix} \eta(k) \\ \vartheta(k) \end{bmatrix} + \begin{bmatrix} 0 & -HRD \end{bmatrix} \begin{bmatrix} v(k) \\ w(k) \end{bmatrix} \end{aligned}$$

Thus η is unobservable from e. Hence, the modes of A are cancelled in the transfer function $T_{e,\eta}$. From (2.4)–(2.7) it follows that

$$\begin{aligned} e(\lambda) &= \left[\frac{M(\lambda)}{d(\lambda)} - \frac{B(\lambda)}{a(\lambda)}\frac{N(\lambda)}{d(\lambda)}\right] v(\lambda) - \left[\frac{B(\lambda)}{a(\lambda)}\right] Dw(\lambda) \\ &= \left[\frac{M(\lambda)a(\lambda) - B(\lambda)N(\lambda)}{a(\lambda)d(\lambda)}\right] v(\lambda) - \left[\frac{B(\lambda)}{a(\lambda)}\right] Dw(\lambda) \end{aligned}$$

Since $d(\lambda) = det(I - A\lambda)$, the cancellation of the modes of A in $T_{e,\eta}$ implies that $M(\lambda)a(\lambda) - B(\lambda)N(\lambda)$ has $d(\lambda)$ as a factor, precisely what (2.13) states. □

We are interested in generalized-observer-like filters, so we will consider only polynomial matrices satisfying (2.13) for some $C(\lambda)$. In this case the estimation error is governed by the equation

$$e(\lambda) = \frac{C(\lambda)}{a(\lambda)} v(\lambda) - \frac{B(\lambda)}{a(\lambda)} Dw(\lambda) \tag{2.15}$$

and the corresponding ARMA model is

$$e(k) = -\sum_{i=1}^{r} a_i e(k-i) + \sum_{i=0}^{r} C_i v(k-i) - \sum_{i=0}^{r} B_i D w(k-i) \tag{2.16}$$

2.3 Equalized Performance Filtering

Now the equalized–performance filtering problem is formally stated.

Problem 1. *Given an integer $r \geq n$ and $\mu > 0$, the aim is to find a filter of the form* (2.6) *of order r satisfying the constraint* (2.13) *and such that $a(\lambda)$ is stable (i.e. all its poles are* outside *the unit circle*[1]*) and*

$$\begin{aligned} &|e(k-j)| \leq \mu, j = 1, 2, \ldots, r \Rightarrow |e(t)| \leq \mu \\ &\text{for all } t \geq k \text{ and all sequences } v, w \in \mathcal{B}\ell^{\infty} \end{aligned} \tag{2.17}$$

The problem above does not explicitly make any assumptions on x_o, the initial conditions of the plant. As we will show later, in section 2.6, if the plant achieves an equalized performance level $\mu < \infty$, then there exist a set of initial conditions $\mathcal{X}_o(\mu)$ such that if $x_o \in \mathcal{X}_o(\mu)$ then $|e(k)| \leq \mu$ for all k. For initial conditions outside this set, the condition will be satisfied after a finite number of steps.

Theorem 1. *An r^{th} order filter of the form* (2.6) *with*

$$\begin{aligned} a(\lambda) &= 1 + a_1\lambda + \cdots + a_r\lambda^r \\ B(\lambda) &= B_0 + B_1\lambda + \cdots + B_r\lambda^r \\ C(\lambda) &= C_0 + C_1\lambda + \cdots + C_r\lambda^r \end{aligned}$$

solves Problem 1 above if and only if

$$\begin{aligned} \mu\|[a_1\ a_2 \ldots a_r]\|_1 &+ \|[C_0\ C_1 \ldots C_r]\|_1 \\ &+ \|B_0 D\ \ldots B_r D\|_1 \leq \mu \end{aligned} \tag{2.18}$$

Proof. See [1]. □

Note that condition (2.18) implies the stability of the filter, since we have that

$$\|[a_1\ a_2 \ldots a_r]\|_1 = \sum_{i=1}^{r} |a_i| = \rho < 1$$

Hence stability is assured.

Filters satisfying (2.18) are r–equalized filters with performance μ. The infimum feasible value of μ

$$\mu_{opt} = \inf \{\mu > 0 : \text{ such that } (2.18) \text{ holds}\}$$

is referred to as the *optimal equalized filtering level.*

[1] since $\lambda = 1/z$

2.4 Optimal Fixed–Order Synthesis

Fixed complexity equalized filters can be efficiently synthesized by combining convex optimization and line search. This is shown in [1] by rewriting (2.13) as the following linear constraint in the variables a_k, B_k and C_k:

$$\mathcal{M}\mathcal{A} - \mathcal{B}\begin{bmatrix} N_0 \\ N_1 \\ \vdots \\ N_l \end{bmatrix} = \mathcal{C}\mathcal{D}$$

where

$$\mathcal{M} \doteq \begin{bmatrix} M_0 & 0 & \dots & 0 & 0 \\ M_1 & M_0 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ M_r & M_{r-1} & \dots & M_1 & M_o \\ \vdots & \vdots & \dots & \vdots & \vdots \\ M_n & M_{n-1} & \dots & M_{n-r+1} & M_{n-r} \\ 0 & M_n & M_{n-1} & \vdots & \vdots \\ \vdots & \vdots & 0 & M_n & M_{n-1} \\ 0 & 0 & \dots & 0 & M_n \end{bmatrix}$$

$$\mathcal{A} \doteq \text{block-diag}\left\{a, a, \dots, a\right\},\ a \doteq \begin{bmatrix} 1 & a_1 & \dots & a_r \end{bmatrix}^T$$

$$\mathcal{B} \doteq \begin{bmatrix} B_0 & 0 & \dots & 0 & 0 \\ B_1 & B_0 & \dots & 0 & 0 \\ : & B_1 & \dots & B_0 & 0 \\ B_r & : & \dots & B_1 & B_0 \\ 0 & B_r & \dots & : & : \\ 0 & : & \dots & B_r & B_{r-1} \\ 0 & 0 & \dots & 0 & B_r \end{bmatrix},\ \mathcal{C} \doteq \begin{bmatrix} C_0 & 0 & \dots & 0 & 0 \\ C_1 & C_0 & \dots & 0 & 0 \\ : & C_1 & \dots & C_0 & 0 \\ C_r & : & \dots & C_1 & C_0 \\ 0 & C_r & \dots & : & : \\ 0 & : & \dots & C_r & C_{r-1} \\ 0 & 0 & \dots & 0 & C_r \end{bmatrix},$$

$$\mathcal{D} \doteq \text{block-diag}\left\{d, d, \dots, d\right\},\ d \doteq \begin{bmatrix} 1 & d_1 & \dots & d_r \end{bmatrix}^T$$

Since for a fixed μ, (2.18) is also convex in a_k, B_k and C_k, it follows that establishing feasibility of (2.13)–(2.18) reduces to a convex problem. Finally, the optimal filter (and its associated optimal filtering error μ_{opt}) can be found via bisection as follows.

Algorithm 1. *0. Select $\mu > 0$ and tolerance δ; set $\mu^- = 0$.*

1. *Solve the feasibility problem (2.13)–(2.18). If it is unfeasible set $\mu = 2\mu$ and go to step 1; else set $\mu^+ = \mu$.*
2. *Solve the feasibility problem for $\mu = (\mu^+ + \mu^-)/2$.*
3. *If it is feasible, set $\mu^+ = \mu$; else set $\mu^- = \mu$.*
4. *If $\mu^+ - \mu^- < \delta$ then set $\mu_{opt} = \mu$ and STOP; else go to step 2.*

2.5 The Multi–Output Case

The results can be extended to the multiple outputs case, $z \in R^s$, by simply considering an array of single–output filters, each of which estimates one of the components of z.

The definition of equalized filtering performance in the multi–output case becomes the following.

Definition 3. *The filter* (2.6) *with error* $z-\hat{z}=e\in R^s$ *is said to achieve a vector equalized performance level* $\mu\doteq\left[\mu_1,\mu_2,\ldots,\mu_s\right]$ *if it is stable and*

$$\begin{array}{l} e(k-j)\in\mathcal{B}\ell^\infty(\mu), j=1,2,\ldots,r\Rightarrow e(k)\in\mathcal{B}\ell^\infty(\mu);\\ \text{for all sequences } v,w\in\mathcal{B}\ell^\infty \end{array} \tag{2.19}$$

Vector equalized performance is equivalent to componentwise scalar equalized performance.

Theorem 2. *A filter* $F\colon y\in\ell_n^\infty\to\hat{z}\in\ell_s^\infty$ *achieves a vector equalized performance level* μ *if and only if each component* $F_i\colon y\in\ell_n^\infty\to\hat{z}^i\in\ell^\infty$ *achieves scalar equalized performance (in the sense of* (2.17)*)* μ_i*, where* $\hat{z}^i$ *denotes the* i^{th} *component of* $\hat{z}$*.*

Proof. See [1]. □

Hence, optimal MIMO filters can be synthesized by simply applying Algorithm 1 componentwise.

2.6 Filter Initialization

We report for completeness the initialization problem from [1]. Given an initial set of r measurements, $\mathbf{y}\doteq\left[y_o,y_1,\ldots,y_{r-1}\right]$ there exist a finite performance level μ and a filter initial condition ξ_o such that the estimation error satisfies $e(k)\in\mathcal{B}\ell^\infty(\mu)$ for all k. Let $\mathcal{K}_o$, T_y and T_z^j denote respectively the r^{th} order Kalman observability matrix of the system (2.3) and the Toeplitz operators mapping v to y and to the j^{th} component of z, e.g.

$$\mathcal{K}_o = \begin{bmatrix} C_y\\ C_yA\\ \vdots\\ C_yA^{r-1}\end{bmatrix}\quad T_y=\begin{bmatrix} 0 & 0 & 0 & \ldots & 0\\ C_yB_v & 0 & 0 & \ldots & 0\\ C_yAB_v & C_yB_v & 0 & \ldots & 0\\ \vdots & \vdots & \ddots & \ddots & \vdots\\ C_yA^{r-2}B_v & C_yA^{r-3}B_v & \ldots & C_yB_v & 0\end{bmatrix}$$

$$T_z^j \doteq \begin{bmatrix} 0 & 0 & 0 & \ldots & 0\\ H(j,:)B_v & 0 & 0 & \ldots & 0\\ H(j,:)AB_v & H(j,:)B_v & 0 & \ldots & 0\\ \vdots & \vdots & \ddots & \ddots & \vdots\\ H(j,:)A^{r-2}B_v & H(j,:)A^{r-3}B_v & \ldots & H(j,:)B_v & 0\end{bmatrix}$$

Then, after r measurements, the filter can be initialized as follows (see Fig. 2.3):

1. For $k=0,\ldots,r-1$ and $j=1,\ldots,s$ compute:

$$\begin{array}{ll} z_k^{j,+} & \doteq \max\limits_{x,v,w} H(j,:)A^{k-1}\mathbf{x}+T_z^j(k,:)\mathbf{v}\\ z_k^{j,-} & \doteq \min\limits_{x,v,w} H(j,:)A^{k-1}\mathbf{x}+T_z^j(k,:)\mathbf{v}\\ & \text{subject to: } \mathbf{x}\in\mathcal{X}_o\\ & \mathbf{y}=T_y\mathbf{v}+\mathcal{D}\mathbf{w},\quad \mathbf{w},\mathbf{v}\in\mathcal{B}\ell_\infty \end{array} \tag{2.20}$$

   where $\mathcal{D}\doteq\text{diag}\{D^T\}$ and where $\mathcal{X}_o$ is a set known to contain the initial condition (if no information is available then $\mathcal{X}_o=R^n$).

2. Define:

$$\begin{array}{rcl} z_k^{j,c} & \doteq & \frac{z_k^{j,+}+z_k^{j,-}}{2},\\ \mu_k^j & \doteq & \frac{1}{2}|z_k^{j,+}-z_k^{j,-}| \end{array} \tag{2.21}$$

3. Let $\mu^{init,j}=\max\limits_{0\le t\le r-1}\{\mu_t^j\}$ and choose a filter initial condition such that the first r filter estimates are $\hat{z}_t=z_t^{j,c}$, $t=0,\ldots,r-1$. This is always feasible, since the order of the filter is precisely r.

If $\mathcal{X}_o$ is convex, then the optimization problem (2.20) is convex. Further, if $\mathcal{X}_o$ is a polytope[2], the problem reduces to LP (*Linear Programming*). Thus z^+, z^-, and z^c above can be found efficiently.

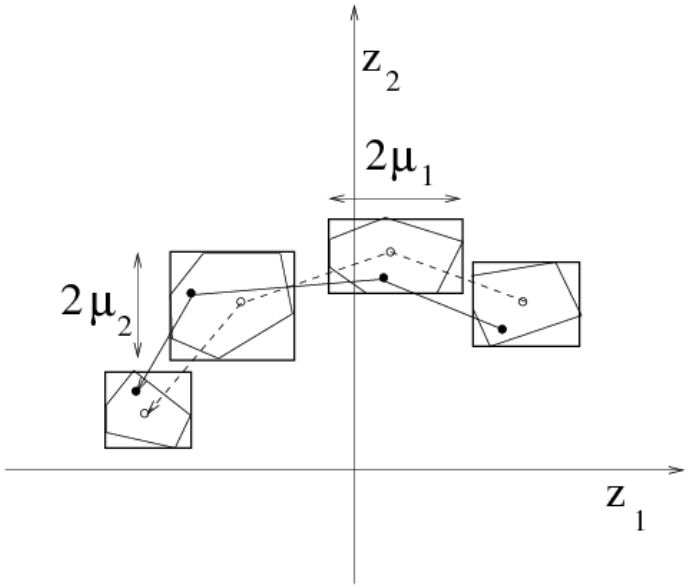


Figure 2.3: The filter initialization (from [1])

Since $\| \begin{bmatrix} a_1 \dots a_r \end{bmatrix} \|_1 < 1$ by construction, if (2.18) holds for some $\tilde{\mu}$, then it also holds for all $\mu \geq \tilde{\mu}$, and so does (2.17). It follows that if $\mu^{init,j} \leq \mu^j_{opt}$ (where μ^j_{opt} is the optimal equalized performance level in (2.18)), then the filter (2.6), with the initialization above, achieves optimal equalized performance level μ^j_{opt} for all $t \geq r$. On the other hand, if $\mu^{init,j} > \mu^j_{opt}$, then the worst case ℓ^∞ estimation error is bounded above by $\mu^{init,j}$ and converges, in a finite number of steps, to μ^j_{opt}.

Theorem 3. *Consider a filter of the form* (2.6)*, with the initialization above. Then, for all* t*,* $|e(t)| \leq \mu^{init}$*. Moreover, given* $\mu \geq \mu_{opt}$ *satisfying* (2.18)*, for any plant and filter initial condition pairs* $\{x_o, \xi_o\}$ *there exists a finite time* $T(x_o, \xi_o, \mu)$ *such that for all* $t > T$*,* $|e(t)| \leq \mu$*.*

Proof. See [1]. □

In the sequel we will be concerned with the steady–state performance of the filter. Therefore we will not consider the initialization problem and the filter will be always initialized at zero.

[2]A polytope is a bounded polyhedron; a polyhedron is the solution set of a finite number of linear equalities and inequalities, thus the intersection of a finite number of halfspaces and hyperplanes: $\mathcal{P} = \{x | a_j^T x \leq b_j, j = 1, \dots, m, c_j^T x = d_j, j = 1, \dots, p\}$.

Chapter 3

Filtering in the Presence of Known Inputs

The purpose of this chapter is to analyse the behaviour of the class of filters developed by Blanchini and Sznaier [1] in a more general context.

Now we will consider systems not only affected by unknown, ℓ^∞ bounded disturbances, but also controlled by an external, known input.

3.1 Problem Statement

Consider an LTI plant subject to ℓ^∞ bounded disturbances, with state space realization

$$\begin{aligned} x(k+1) &= Ax(k) + B_v v(k) + Eu(k) \\ z(k) &= Hx(k) \\ y(k) &= C_y x(k) + Dw(k) \end{aligned} \tag{3.1}$$

or with λ–transform representation

$$z(\lambda) = \frac{M(\lambda)}{d(\lambda)} v(\lambda) + \frac{\Psi(\lambda)}{d(\lambda)} u(\lambda) \tag{3.2}$$

$$y(\lambda) = \frac{N(\lambda)}{d(\lambda)} v(\lambda) + \frac{\Phi(\lambda)}{d(\lambda)} u(\lambda) + Dw(\lambda) \tag{3.3}$$

where $z \in \mathbb{R}^s$ is the output to be estimated, $y \in \mathbb{R}^q$ are the measurements available to the filter, $u \in \mathbb{R}^m$ is the known (control) input, $v \in \mathbb{R}^p$ is process noise and $w \in \mathbb{R}^q$ is measurement noise; besides $d(\lambda) = det(I - \lambda A)$.

Again, the purpose is to obtain an estimation $\hat{z}$ of z such that $e(\lambda) = z(\lambda) - \hat{z}(\lambda)$ is as small as possible.

To this aim, we look for a filter of the form

$$\hat{z}(\lambda) = \frac{B(\lambda)}{a(\lambda)} y(\lambda) + \frac{\Gamma(\lambda)}{a(\lambda)} u(\lambda), \tag{3.4}$$

where B is the same polynomial matrix and a the same polynomial that form the filter (2.6), obtained with the same procedure described in [1] and reported in the previous chapter.

We consider filters belonging to the class of generalized Luenberger observers, i.e. systems of the form

$$\xi(k+1) = P\xi(k) + Ly(k) + TEu(k) \tag{3.5}$$

$$\hat{x}(k) = Q\xi(k) + Ry(k) \tag{3.6}$$

$$\hat{z}(k) = H\hat{x}(k) \tag{3.7}$$

with P stable and $\hat{x}(k) - x(k) \to 0$ as $k \to \infty$, when $w(k) \equiv 0$ and $v(k) \equiv 0$.

As Lemma 2 in the previous chapter claims, the system (3.5)–(3.7) is a generalized Luenberger state observer for (3.1) if and only if P is stable and there exists a full column rank matrix T such that

$$\begin{aligned} TA - LC_y &= PT \quad (3.8) \\ QT + RC_y &= I \quad (3.9) \end{aligned}$$

We show that, if P is stable and (3.8)–(3.9) are valid, $\hat{x}(k) - x(k) \to 0$ as $k \to \infty$, when $w(k) \equiv 0$ and $v(k) \equiv 0$.

$$\begin{aligned} [Tx - \xi](k+1) &= TAx(k) + TB_v v(k) + TEu(k) - P\xi(k) - Ly(k) - TEu(k) \\ &= TAx(k) + TB_v v(k) - P\xi(k) - LC_y x(k) - LDw(k) \\ &= (TA - LC_y)x(k) + TB_v v(k) - P\xi(k) - LDw(k) \\ &= PTx(k) - P\xi(k) + TB_v v(k) - LDw(k) \\ &= P[Tx - \xi](k) + TB_v v(k) - LDw(k) \end{aligned}$$

Since P is stable, when $w(k) \equiv 0$ and $v(k) \equiv 0$, $[Tx - \xi](k) \to 0$ as $k \to \infty$.

$$\begin{aligned} \hat{x}(k) - x(k) &= Q\xi(k) + R[C_y x(k) + Dw(k)] - Ix(k) \\ &= Q\xi(k) + [RC_y - I]x(k) + RDw(k) \\ &= Q\xi(k) - QTx(k) + RDw(k) \\ &= -Q[Tx - \xi](k) + RDw(k) \end{aligned}$$

Thus, when $w(k) \equiv 0$ and $v(k) \equiv 0$, as $k \to \infty$ $\hat{x}(k) - x(k) \to 0$, because $[Tx - \xi](k) \to 0$.

3.2 The Case $B_v = E$

We assume $B_v = E$ (and therefore $u \in \mathbb{R}^m$ with $m = p$, since $v \in \mathbb{R}^p$) which means that the known input u and the process noise v enter through the same channel.

The state space realization becomes

$$\begin{aligned} x(k+1) &= Ax(k) + B_v[v(k) + u(k)] \\ z(k) &= Hx(k) \\ y(k) &= C_y x(k) + Dw(k) \end{aligned}$$

and in terms of λ–transform we have

$$\begin{aligned} z(\lambda) &= \frac{M(\lambda)}{d(\lambda)}[v(\lambda) + u(\lambda)] \\ y(\lambda) &= \frac{N(\lambda)}{d(\lambda)}[v(\lambda) + u(\lambda)] + Dw(\lambda) \end{aligned}$$

As a preliminary step we put $\Gamma(\lambda) = 0$ and consider a filter of the form

$$\hat{z}(\lambda) = \frac{B(\lambda)}{a(\lambda)} y(\lambda),$$

which is exactly the same of (2.6), just to evidence the effect of the known input. The scheme of the new situation is illustrated in Fig. 3.1.

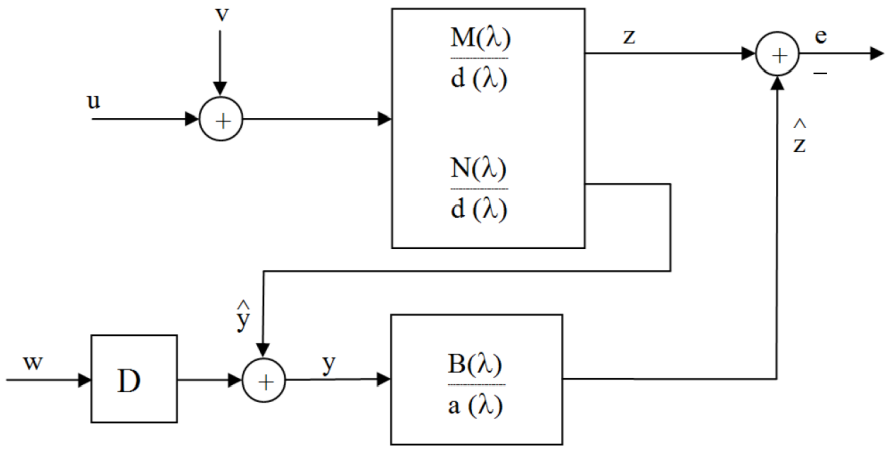


Figure 3.1: Block scheme when a known input is present and $B_v = E$.

The estimation error is

$$\begin{aligned} e(\lambda) &= z(\lambda) - \hat{z}(\lambda) \\ &= \frac{M(\lambda)}{d(\lambda)}[v(\lambda) + u(\lambda)] - \frac{B(\lambda)}{a(\lambda)} y(\lambda) \\ &= \frac{M(\lambda)}{d(\lambda)}[v(\lambda) + u(\lambda)] - \frac{B(\lambda)}{a(\lambda)} \left\{ \frac{N(\lambda)}{d(\lambda)}[v(\lambda) + u(\lambda)] + Dw(\lambda) \right\} \\ &= \left[\frac{M(\lambda)}{d(\lambda)} - \frac{B(\lambda)}{a(\lambda)} \frac{N(\lambda)}{d(\lambda)} \right] [v(\lambda) + u(\lambda)] - \frac{B(\lambda)}{a(\lambda)} Dw(\lambda) \\ &= \frac{M(\lambda)a(\lambda) - B(\lambda)N(\lambda)}{a(\lambda)d(\lambda)} [v(\lambda) + u(\lambda)] - \frac{B(\lambda)}{a(\lambda)} Dw(\lambda). \end{aligned}$$

Since B and a are computed as in the previous chapter, the constraint (2.13)

$$M(\lambda)a(\lambda) - B(\lambda)N(\lambda) = C(\lambda)d(\lambda),$$

still holds.

Therefore the estimation error

$$\begin{aligned} e(\lambda) &= \frac{C(\lambda)}{a(\lambda)}[v(\lambda) + u(\lambda)] - \frac{B(\lambda)}{a(\lambda)} Dw(\lambda) \\ &= \frac{C(\lambda)}{a(\lambda)} v(\lambda) - \frac{B(\lambda)}{a(\lambda)} Dw(\lambda) + \frac{C(\lambda)}{a(\lambda)} u(\lambda) \end{aligned}$$

is affected by u (which can be arbitrarily large).

To improve the filter, it is crucial to remember that the control input u is known, so it is possible to reduce the estimation error. Adopting the more general expression of the filter

$$\hat{z}(\lambda) = \frac{B(\lambda)}{a(\lambda)} y(\lambda) + \frac{\Gamma(\lambda)}{a(\lambda)} u(\lambda),$$

and assuming $\Gamma(\lambda) = C(\lambda)$ we obtain

$$\hat{z}(\lambda) = \frac{B(\lambda)}{a(\lambda)} y(\lambda) + \frac{C(\lambda)}{a(\lambda)} u(\lambda) \tag{3.10}$$

which corresponds to the scheme shown in Fig. 3.2.

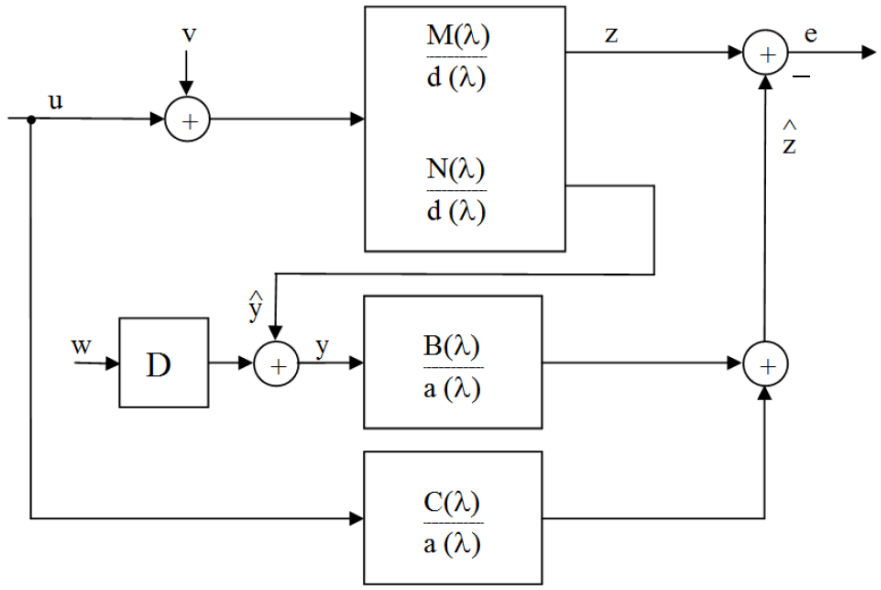


Figure 3.2: The filtering scheme in presence of a known input with $B_v = E$.

The estimation error with this new filter is

$$\begin{aligned}
e(\lambda) &= z(\lambda) - \hat{z}(\lambda) \\
&= \frac{M(\lambda)}{d(\lambda)}[v(\lambda)+u(\lambda)] - \frac{B(\lambda)}{a(\lambda)}y(\lambda) - \frac{C(\lambda)}{a(\lambda)}u(\lambda) \\
&= \frac{M(\lambda)}{d(\lambda)}[v(\lambda)+u(\lambda)] - \frac{B(\lambda)}{a(\lambda)}\left\{\frac{N(\lambda)}{d(\lambda)}[v(\lambda)+u(\lambda)] + Dw(\lambda)\right\} - \frac{C(\lambda)}{a(\lambda)}u(\lambda) \\
&= \left[\frac{M(\lambda)}{d(\lambda)} - \frac{B(\lambda)}{a(\lambda)}\frac{N(\lambda)}{d(\lambda)}\right][v(\lambda)+u(\lambda)] - \frac{B(\lambda)}{a(\lambda)}Dw(\lambda) - \frac{C(\lambda)}{a(\lambda)}u(\lambda) \\
&= \frac{M(\lambda)a(\lambda) - B(\lambda)N(\lambda)}{a(\lambda)d(\lambda)}[v(\lambda)+u(\lambda)] - \frac{B(\lambda)}{a(\lambda)}Dw(\lambda) - \frac{C(\lambda)}{a(\lambda)}u(\lambda) \\
&= \frac{C(\lambda)}{a(\lambda)}[v(\lambda)+u(\lambda)] - \frac{B(\lambda)}{a(\lambda)}Dw(\lambda) - \frac{C(\lambda)}{a(\lambda)}u(\lambda) \\
&= \frac{C(\lambda)}{a(\lambda)}v(\lambda) - \frac{B(\lambda)}{a(\lambda)}Dw(\lambda),
\end{aligned}$$

which is the same estimation equation obtained in the previous chapter *in absence of* u.

Comments

We have seen that, to estimate the output z of a system affected by bounded noise and in presence of a known input, it is possible to use the filter

$$\hat{z}(\lambda) = \frac{B(\lambda)}{a(\lambda)}y(\lambda) + \frac{C(\lambda)}{a(\lambda)}u(\lambda),$$

where a, B and C are computed by solving the convex optimization feasibility problem described in [1], as it was reported in the previous chapter.

Since the estimation error attained by this filter, also in presence of control (or more in general of a known input), is the same that was achieved under the assumptions of [1], it is likely to be a good filter. It will be tested upon several numerical examples in the following part of this thesis, dedicated to simulations.

Yet it is important to remember that the simulations are performed via *numerical* algorithms, with *finite* precision. So the exact cancellation that occurs in theory is indeed

not perfect and the influence of u on the estimation error is *not perfectly* eliminated. It can be seen that, when the amplitude of u increases, also the estimation error becomes larger. Due to numerical effects, u still affects the estimation error, even though its influence is much smaller than the influence of disturbances, when the degree of magnitude is the same, because u is known.

A similar effect will be seen later, when an unstable system will be filtered and, due to an imperfect pole–zero cancellation, the estimation error will diverge.

3.3 The Case $B_v \neq E$

If we consider (3.1) in the most general case, the λ–transform representation is

$$\begin{aligned} z(\lambda) &= \frac{M(\lambda)}{d(\lambda)}v(\lambda) + \frac{\Psi(\lambda)}{d(\lambda)}u(\lambda) \\ y(\lambda) &= \frac{N(\lambda)}{d(\lambda)}v(\lambda) + \frac{\Phi(\lambda)}{d(\lambda)}u(\lambda) + Dw(\lambda) \end{aligned}$$

and, if the form of the filter is

$$\hat{z}(\lambda) = \frac{B(\lambda)}{a(\lambda)}y(\lambda),$$

the estimation error can be calculated as

$$\begin{aligned} e(\lambda) &= z(\lambda) - \hat{z}(\lambda) \\ &= \frac{M(\lambda)}{d(\lambda)}v(\lambda) + \frac{\Psi(\lambda)}{d(\lambda)}u(\lambda) - \frac{B(\lambda)}{a(\lambda)}\left[\frac{N(\lambda)}{d(\lambda)}v(\lambda) + \frac{\Phi(\lambda)}{d(\lambda)}u(\lambda) + Dw(\lambda)\right] \\ &= \left[\frac{M(\lambda)}{d(\lambda)} - \frac{B(\lambda)}{a(\lambda)}\frac{N(\lambda)}{d(\lambda)}\right]v(\lambda) - \frac{B(\lambda)}{a(\lambda)}Dw(\lambda) + \left[\frac{\Psi(\lambda)}{d(\lambda)} - \frac{B(\lambda)}{a(\lambda)}\frac{\Phi(\lambda)}{d(\lambda)}\right]u(\lambda) \\ &= \frac{C(\lambda)}{a(\lambda)}v(\lambda) - \frac{B(\lambda)}{a(\lambda)}Dw(\lambda) + \frac{\Psi(\lambda)a(\lambda) - B(\lambda)\Phi(\lambda)}{a(\lambda)d(\lambda)}u(\lambda) \end{aligned}$$

since it is still valid the constraint (2.13). The situation is represented in Fig. 3.3.

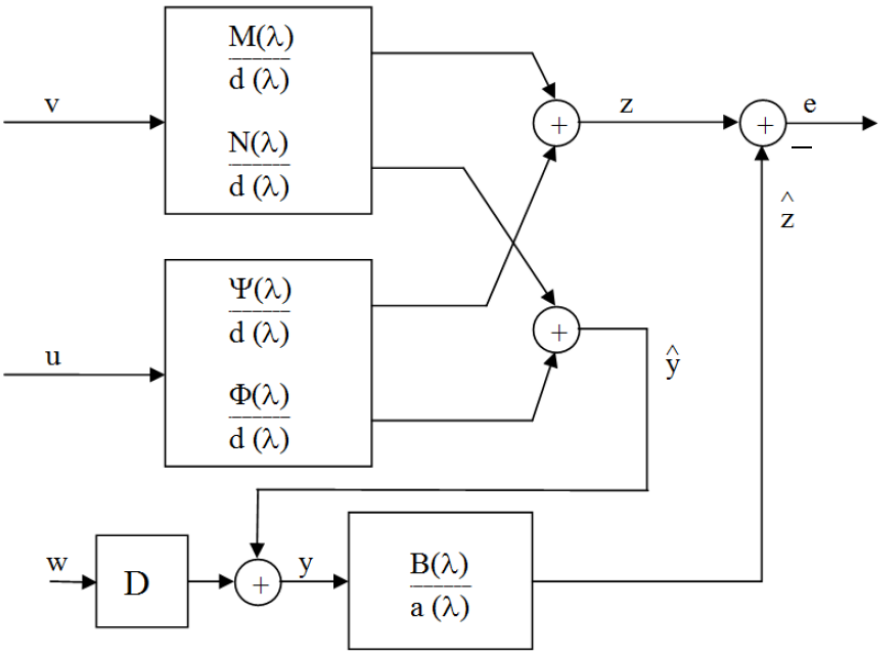


Figure 3.3: The filtering scheme in presence of a known input with $B_v \neq E$.

Then we can adopt the same trick we used in the previous section (when $B_v = E$) by adding a term in u to the filter as follows

$$\hat{z}(\lambda) = \frac{B(\lambda)}{a(\lambda)} y(\lambda) + \frac{\Psi(\lambda)a(\lambda) - B(\lambda)\Phi(\lambda)}{a(\lambda)d(\lambda)} u(\lambda)$$

Clearly we should ensure the stability of the additional component of the filter (the system is not assumed to be stable). Essentially, we must impose that $\Psi(\lambda)a(\lambda) - B(\lambda)\Phi(\lambda)$ has $d(\lambda)$ as a factor:

$$\Psi(\lambda)a(\lambda) - B(\lambda)\Phi(\lambda) = \Lambda(\lambda)d(\lambda)$$

for a certain polynomial matrix $\Lambda(\lambda)$. This requires imposing another constraint on a, B and C and solving the convex optimization feasibility problem with an additional constraint.

Thus the filter obtained through the procedure described in [1] by Blanchini and Sznaier is no more suitable in this different context. Since the purpose of this thesis is to analyse that particular filter, from now on we will take into consideration only examples of the case $B_v = E$.

Chapter 4

Discrete–Time Kalman Filter

4.1 The Kalman Filtering Algorithm

Consider an LTI plant with state space realization

$$\begin{aligned} x(k+1) &= Ax(k) + Bv(k) \\ z(k) &= Hx(k) \\ y(k) &= Cx(k) + w(k) \end{aligned}$$

where $z \in \mathbb{R}^s$ is the output to be estimated, $y \in \mathbb{R}^q$ are the measurements available to the filter, $v \in \mathbb{R}^p$ is process noise and $w \in \mathbb{R}^q$ is measurement noise. We aim at obtaining a recursive algorithm that yields an estimate $\hat{x}$ of the process state.

Assume that w and v are stochastic processes and

1. $E[v(k)] = 0$
2. $E[v(k)v(h)^T] = Q(k)\delta(k-h)$, with $Q(k) > 0$
3. $E[x(k_0)] = \overline{x}_0$
4. $E[(x(k_0) - \overline{x}_0)(x(k_0) - \overline{x}_0)^T] = Var[x(k_0)] = P_0$
5. $E[w(k)] = 0$
6. $E[w(k)w(h)^T] = R(k)\delta(k-h)$, with $R(k) > 0$
7. $E[w(k)v(h)^T] = 0$ for every k, h
8. $E[w(k)x(h)^T] = 0$ for every k, h
9. $E[v(k)x(h)^T] = 0$ for $k \geq h$

These hypotheses mean that: v, w are white noises; the statistics of x_0 is known; v, w, x are uncorrelated.

We consider filters belonging to the class of Luenberger observers, i.e. we look for an estimate of the state generated by the plant

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

where $L(k) \in \mathbb{R}^{n \times q}$ is the matrix we want to determine.

The estimation error is

$$\begin{aligned} & e(k+1) = x(k+1) - \hat{x}(k+1) \\ & = A(k)x(k) + B(k)v(k) - [A(k) - L(k)C(k)]\hat{x}(k) - L(k)[C(k)x(k) + w(k)] \\ & = [A(k) - L(k)C(k)][x(k) - \hat{x}(k)] + B(k)v(k) - L(k)w(k) \\ & = [A(k) - L(k)C(k)]e(k) + B(k)v(k) - L(k)w(k) \end{aligned}$$

We want to find the sequence of matrices $L(k)$ that minimizes the quantity

$$E[\|x(k) - \hat{x}(k)\|^2] = trE[(x(k) - \hat{x}(k))(x(k) - \hat{x}(k))^T] \ (^1).$$

Remark 1. *The equality holds because* $\left\| \ [\kappa_1 \ \kappa_2 \ \kappa_3 \ldots \kappa_n]^T \right\|^2 = \kappa_1^2 + \kappa_2^2 + \kappa_3^2 + \cdots + \kappa_n^2$ *and*

$$\begin{bmatrix} \kappa_1 \\ \kappa_2 \\ \kappa_3 \\ \vdots \\ \kappa_n \end{bmatrix} \begin{bmatrix} \kappa_1 & \kappa_2 & \kappa_3 & \ldots & \kappa_n \end{bmatrix} = \begin{bmatrix} \kappa_1^2 & \kappa_1\kappa_2 & \kappa_1\kappa_3 & \ldots & \kappa_1\kappa_n \\ \kappa_1\kappa_2 & \kappa_2^2 & \kappa_2\kappa_3 & \ldots & \kappa_2\kappa_n \\ \kappa_1\kappa_3 & \kappa_2\kappa_3 & \kappa_3^2 & \ldots & \kappa_3\kappa_n \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \kappa_1\kappa_n & \kappa_2\kappa_n & \kappa_3\kappa_n & \ldots & \kappa_n^2 \end{bmatrix}$$

whose trace is exactly $\kappa_1^2 + \kappa_2^2 + \kappa_3^2 + \cdots + \kappa_n^2$.

We define $P(k) = E[(x(k) - \hat{x}(k))(x(k) - \hat{x}(k))^T]$ and so we have $P(k_0) = P_0$. Initially, the best estimate is $\hat{x}(k_0) = \overline{x}_0$. Recursively, given $P(k)$, $P(k+1)$ is determined as

$$[A(k) - L(k)C(k)]P(k)[A(k) - L(k)C(k)]^T + B(k)Q(k)B^T(k) + L(k)R(k)L^T(k)$$

We choose $P(k+1)$ so as we minimize $tr[P(k)]$:

$$\begin{aligned} &\min_L tr[(A - LC)P(A - LC)^T + BQB^T + LRL^T)] = \\ &\min_L tr[APA^T] - 2tr[APC^TL^T] + tr[LCPC^TL^T] + tr[BQB^T] + tr[LRL^T] \end{aligned}$$

Remark 2. $(A - LC)P(A - LC)^T = (A - LC)(PA^T - PC^TL^T) = APA^T - LCPA^T - APC^TL^T + LCPC^TL^T$. *Since* $LCPA^T = (APC^TL^T)^T$, *because* P *is symmetric, their trace is the same: the trace of a matrix is always equal to that of its transpose.*

Differentiating with respect to L and equating to zero, we obtain

$$2L(CPC^T + R) - 2APC^T = 0$$

and thus

$$L = APC^T(CPC^T + R)^{-1}$$

When we substitute the expression obtained for L into the expression of $P(k+1)$, we have

$$\begin{aligned} &(A - LC)P(A - LC)^T + BQB^T + LRL^T \\ &= APA^T + APC^T(CPC^T + R)^{-1}CPC^T(CPC^T + R)^{-1}CPA^T \\ &-APC^T(CPC^T + R)^{-1}CPA^T - APC^T(CPC^T + R)^{-1}CPA^T \\ &+BQB^T + APC^T(CPC^T + R)^{-1}R(CPC^T + R)^{-1}CPA^T \\ &= APA^T - APC^T(CPC^T + R)^{-1}CPA^T + BQB^T \end{aligned}$$

Thus the optimal recursive estimate is given by the process

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

where

$$L(k) = A(k)P(k)C^T(k)[C(k)P(k)C^T(k) + R(k)]^{-1}$$

and $P(k)$ is recursively given by

$$\begin{aligned} P(k+1) \ &= \ A(k)P(k)A^T(k) - A(k)P(k)C^T(k)[C(k)P(k)C^T(k) + R(k)]^{-1}C(k)P(k)A^T(k) \\ &+ \ B(k)Q(k)B^T(k), \end{aligned}$$

with the initial conditions $P(k_0) = P_0$ and $x(k_0) = \overline{x}_0$.

[1] $trS = \sum_i S_{ii}$ is the trace of matrix S, i.e. the sum of its diagonal elements

Remark 3. *If $v(k)$ and $w(k)$ do not have zero average, but have a known average $\overline{v}(k)$ and $\overline{w}(k)$, respectively, the filter is given by*

$$\hat{x}(k+1) = [A(k) - L(k)C(k)]\hat{x}(k) + L(k)y(k) - L(k)\overline{w}(k) + B(k)\overline{v}(k),$$

so that we can exploit the results previously obtained, since the estimation error $x(k+1) - \hat{x}(k+1)$ is

$$[A(k) - L(k)C(k)][x(k) - \hat{x}(k)] + B(k)[v(k) - \overline{v}(k)] - L(k)[w(k) - \overline{w}(k)]$$

If we have also a known input u, for instance a control signal, the plant is

$$\begin{aligned} x(k+1) &= Ax(k) + Bv(k) + Eu(k) \\ z(k) &= Hx(k) \\ y(k) &= Cx(k) + w(k) \end{aligned}$$

where $z \in \mathbb{R}^s$ is the output to be estimated, $y \in \mathbb{R}^q$ are the measurements available to the filter, $u \in \mathbb{R}^m$ is the known input, $v \in \mathbb{R}^p$ is process noise and $w \in \mathbb{R}^q$ is measurement noise.

All the reasoning of the previous section can be repeated similarly: it is sufficient to consider

$$\hat{x}(k+1) = [A(k) - L(k)C(k)]\hat{x}(k) + L(k)y(k) + E(k)u(k),$$

so that the estimation error $x(k+1) - \hat{x}(k+1)$ is

$$\begin{aligned} &A(k)x(k) + B(k)v(k) + E(k)u(k) - [A(k) - L(k)C(k)]\hat{x}(k) \\ &-L(k)[C(k)x(k) + w(k)] - E(k)u(k) \\ &= [A(k) - L(k)C(k)][x(k) - \hat{x}(k)] + B(k)v(k) - L(k)w(k) \end{aligned}$$

as in the previous case.

Remark 4. *The estimate $\hat{x}(k+1)$ provided by the Kalman filter can be obtained by means of a simple equation that involves the previous estimate $\hat{x}(k)$ and the new measurement $y(k)$, but does not involve any of the past measurements $y(0), y(1), \ldots, y(k-1)$.*

4.2 Steady–State Kalman Filtering Algorithm

We have obtained in the previous section that

$$P(k+1) = A(k)P(k)A^T(k) - L(k)C(k)P(k)A^T(k) + B(k)Q(k)B^T(k),$$

with the initial condition $P(k_0) = P_0$.

When all the system matrices are constant, under suitable assumption, $P(k+1)$ tends to a positive definite symmetric matrix P that solves the discrete–time algebraic Riccati equation

$$P = A[P - PC^T(CPC^T + R)^{-1}CP]A^T + BQB^T$$

or

$$P = APA^T - APC^T(CPC^T + R)^{-1}CPA^T + BQB^T,$$

assuming observability of the pair (A, C) and controllability of the pair (A, B). The asymptotic Kalman filter is given by

$$\hat{x}(k+1) = (A - LC)\hat{x}(k) + Ly(k) = A\hat{x}(k) + APC^T(CPC^T + R)^{-1}[y(k) - C\hat{x}(k)],$$

where

$$L = APC^T(CPC^T + R)^{-1}.$$

Stability Aspects

To study now the stability properties of the steady–state form of the Kalman filter, consider the estimation error

$$e(k+1) = x(k+1) - \hat{x}(k+1) = (A - LC)e(k) + Bv(k) - Lw(k)$$

It is important that the error equation represents a stable system, i.e. the matrix $A - LC$ has eigenvalues strictly within the unit circle. This follows, under the observability and controllability assumptions, by taking into account that P is the unique positive definite symmetric solution of the discrete–time algebraic Riccati equation

$$P = A[P - PC^T(CPC^T + R)^{-1}CP]A^T + BQB^T.$$

As it was previously shown, this equation can be rewritten in the form

$$(A - LC)P(A - LC)^T - P = -BQB^T - LRL^T,$$

which is a discrete–time Lyapunov equation. Since $BQB^T + LRL^T$ is symmetric and positive definite and the solution P is positive definite, the matrix $A - LC$ is asymptotically stable and the sequence $\{e_k\}$, generated by the system $e(k+1) = (A - LC)e(k) + Bv(k) - Lw(k)$, tends to zero whenever the vectors $v(k)$ and $w(k)$ are identically zero for all k.

4.3 Estimating z or x?

In the following chapters, the discrete–time Kalman filter will be compared with the equalized filter obtained via convex optimization ([1]). For a fair comparison we will consider $R = D^2$ and $Q = 1$. In this section we deal with the following issue. Since our goal is to estimate z, we need to focus our attention on the estimated output $\hat{z}$, which is assumed to be a scalar ($s = 1$), rather than on the estimated state.

Since $z(k) = Hx(k)$, an estimate of z, based upon the estimate of x, is achieved by

$$\hat{z}(k) = H\hat{x}(k).$$

If only noise is present

$$\begin{aligned} x(k+1) &= Ax(k) + Bv(k) \\ z(k) &= Hx(k) \\ y(k) &= Cx(k) + Dw(k) \end{aligned}$$

the Kalman filter is

$$\begin{aligned} \hat{x}(k+1) &= (A - LC)\hat{x}(k) + Ly(k) \\ \hat{z}(k) &= H\hat{x}(k) \end{aligned}$$

where

$$L = APC^T(CPC^T + D^2)^{-1}$$

and P is the solution of the discrete–time algebraic Riccati equation

$$P = APA^T + APC^T(CPC^T + D^2)^{-1}CPA^T + BB^T.$$

In presence of a known input the system is, instead,

$$\begin{aligned} x(k+1) &= Ax(k) + Bv(k) + Eu(k) \\ z(k) &= Hx(k) \\ y(k) &= Cx(k) + Dw(k) \end{aligned}$$

and the Kalman filter has to be taken as

$$\begin{aligned} \hat{x}(k+1) &= (A - LC)\hat{x}(k) + Ly(k) + Eu(k) \\ \hat{z}(k) &= H\hat{x}(k) \end{aligned}$$

where L and P are the same matrices of the previous case.

The Kalman filter optimizes the trace of $E[(x-\hat{x})(x-\hat{x})^T]$ and does not privilege any particular direction. So, if we look for an optimized estimate of x, will we find an optimized estimate also of z, which is a particular linear combination of the components of x? Indeed, since we are interested in z, the relevant estimation error is not $x-\hat{x}$, but $z-\hat{z}=H(x-\hat{x})$.

So, instead of considering $E[(x-\hat{x})(x-\hat{x})^T]$, we may consider $E[(H(x-\hat{x}))((x-\hat{x})^T H^T)] = E[((x-\hat{x})^T H^T)(H(x-\hat{x}))] = E[(x-\hat{x})^T H^T H(x-\hat{x})]$. To avoid singularity problems, we add a term $\nu^2 I$, where ν^2 is a positive and very small number. This allows the matrix to become positive definite. We have

$$\begin{aligned} E[(x-\hat{x})^T(H^TH+\nu^2 I)(x-\hat{x})] &= E[(x-\hat{x})^T\Omega^2(x-\hat{x})] \\ &= E[(x-\hat{x})^T\Omega\,\Omega(x-\hat{x})] \end{aligned}$$

To analyse this situation, we perform a state transformation of the system in order to exploit the previous theory. Consider the new variable $\chi=\Omega x$ and note that $E[(\chi-\hat{\chi})(\chi-\hat{\chi})^T] = E[(\Omega(x-\hat{x}))((x-\hat{x})^T\Omega^T)] = E[(x-\hat{x})^T\Omega^T\Omega(x-\hat{x})]$ and $\Omega^T=\Omega$ since Ω is symmetric. We apply the transformation

$$\begin{aligned} \chi(k) &= \Omega x(k) \\ x(k) &= \Omega^{-1}\chi(k) \end{aligned}$$

to obtain

$$\begin{array}{lcl} x(k+1)=Ax(k)+Bv(k) & & \Omega^{-1}\chi(k+1)=A\Omega^{-1}\chi(k)+Bv(k) \\ z(k)=Hx(k) & \Rightarrow & z(k)=H\Omega^{-1}\chi(k) \\ y(k)=Cx(k)+Dw(k) & & y(k)=C\Omega^{-1}\chi(k)+Dw(k) \\ & & \chi(k+1)=\Omega A\Omega^{-1}\chi(k)+\Omega Bv(k) \\ & \Rightarrow & z(k)=H\Omega^{-1}\chi(k) \\ & & y(k)=C\Omega^{-1}\chi(k)+Dw(k) \end{array}$$

The system has the new representation

$$\begin{aligned} \chi(k+1) &= A_r\chi(k)+B_rv(k) \\ z(k) &= H_r\chi(k) \\ y(k) &= C_r\chi(k)+D_rw(k) \end{aligned}$$

where the matrices have the following expressions

$$\begin{aligned} A_r &= \Omega A\Omega^{-1} \\ B_r &= \Omega B \\ H_r &= H\Omega^{-1} \\ C_r &= C\Omega^{-1} \\ D_r &= D \end{aligned}$$

Of course matrix D has not changed, since the relation between $y(k)$ and $w(k)$ has not been transformed. Therefore we can obtain the standard discrete–time Kalman filter for the transformed system (with $R=D^2$ and $Q=1$).

When disturbances only are present, we have

$$\begin{aligned} \hat{x}(k+1) &= (A_r-L_rC_r)\hat{x}(k)+L_ry(k) \\ \hat{z}(k) &= H_r\hat{x}(k) \end{aligned}$$

where

$$L_r = A_rP_rC_r^T(C_rP_rC_r^T+D^2)^{-1}$$

and P_r is the solution of the discrete–time algebraic Riccati equation

$$P_r = A_rP_rA_r^T + A_rP_rC_r^T(C_rP_rC_r^T+D^2)^{-1}C_rP_rA_r^T + B_rB_r^T$$

while, *when also a known signal u is present*, we have

$$\begin{aligned} \hat{x}(k+1) &= (A_r - L_r C_r)\hat{x}(k) + L_r y(k) + E_r u(k) \\ \hat{z}(k) &= H_r \hat{x}(k) \end{aligned}$$

where L_r and P_r are the same matrices of the previous case and

$$E_r = \Omega E.$$

Question: Is the Kalman filter designed for the transformed previous system, thought with the aim of estimating z, better than the standard filter?

The answer to this question is negative, because we can easily show that the transfer function from the noises to $\hat{z}$ achieved by means of the Kalman filter is invariant under state transformations[2].

The discrete–time Riccati equation for P_r

$$P_r = A_r P_r A_r^T + A_r P_r C_r^T (C_r P_r C_r^T + D^2)^{-1} C_r P_r A_r^T + B_r B_r^T$$

is equivalent to

$$\begin{aligned} &P_r = \Omega A \Omega^{-1} P_r \Omega^{-1} A^T \Omega + \\ &\Omega A \Omega^{-1} P_r \Omega^{-1} C^T (C \Omega^{-1} P_r \Omega^{-1} C^T + D^2)^{-1} C \Omega^{-1} P_r \Omega^{-1} A^T \Omega + \Omega B B^T \Omega, \end{aligned}$$

since Ω is symmetric, and thus

$$\begin{aligned} &\Omega^{-1} P_r \Omega^{-1} = A \Omega^{-1} P_r \Omega^{-1} A^T + \\ &A \Omega^{-1} P_r \Omega^{-1} C^T (C \Omega^{-1} P_r \Omega^{-1} C^T + D^2)^{-1} C \Omega^{-1} P_r \Omega^{-1} A^T + B B^T. \end{aligned}$$

So $P = \Omega^{-1} P_r \Omega^{-1}$ and $P_r = \Omega P \Omega$, which is an equivalence transformation for quadratic forms. Besides

$$\begin{aligned} L_r &= A_r P_r C_r^T (C_r P_r C_r^T + D^2)^{-1} \\ &= \Omega A \Omega^{-1} \Omega P \Omega \Omega^{-1} C^T (C \Omega^{-1} \Omega P \Omega \Omega^{-1} C^T + D^2)^{-1} \\ &= \Omega A P C^T (C P C^T + D^2)^{-1} = \Omega L, \end{aligned}$$

then also L transforms as B does.

This justifies the fact that in our experiments we will always take into consideration the standard Kalman filter for comparison with the equalized one, without considering weights on the state variables.

[2] this result would follow by duality from the fact that the LQ regulator is optimal for any initial condition

Part II

Examples and Applications

Chapter 5

General Procedure

This second part is dedicated to simulation and analysis of the equalized filter, applied to some specific examples.

Given a system characterized by a process disturbance $|v(t)| \leq \gamma$, a measurement noise $|w(t)| \leq \beta$, disturbance to estimated output (v to z) discrete–time transfer function (expressed by means of λ–transform)

$$\frac{m_0 + m_1\lambda + m_2\lambda^2 + \cdots + m_\nu\lambda^\nu}{d_0 + d_1\lambda + d_2\lambda^2 + \cdots + d_\nu\lambda^\nu} \text{ with } d_0 = 1$$

and disturbance to measured output (v to y) discrete–time transfer function

$$\frac{n_0 + n_1\lambda + n_2\lambda^2 + \cdots + n_\nu\lambda^\nu}{d_0 + d_1\lambda + d_2\lambda^2 + \cdots + d_\nu\lambda^\nu} \text{ with } d_0 = 1,$$

the aim is to synthesize a filter of order $r \geq \nu$

$$\frac{b_0 + b_1\lambda + b_2\lambda^2 + \cdots + b_r\lambda^r}{a_0 + a_1\lambda + a_2\lambda^2 + \cdots + a_r\lambda^r}$$

and to test it with numerical simulations.

Simulation software has been implemented in MATLAB code to study the behaviour of the filter in the considered examples. Each program is composed by three parts:

- a data file that sets the parameters;
- a routine that computes the filter by using an interactive function;
- a simulation program that generates the system response to random disturbances and produces the Bode diagram of the filter and two graphs that show z versus $\hat{z}$ and the estimation error $e = z - \hat{z}$.

The following variables have to be initialized:

- the parameters of the system (state space realization or transfer function)
- the sampling time for discretization
- the values of β and γ
- the order of the filter
- a positive trial value for μ (the interactive program finds μ_{opt} after some iterations)
- the parameters for input signal generation
- the simulation interval

In the following sections the steps that lead to filter synthesis and simulation are reported and discussed.

5.1 Input Data Generation

In all our examples we consider continuous–time systems, which represent real situations in the physical world. Then discretization is necessary. The program must

- generate the continuous–time system matrices from the inserted data;
- convert the continuous–time system to its discrete–time equivalent;
- compute the vectors m, n, d, that contain the coefficients of numerators and denominator of the discrete–time transfer functions.

5.2 Interactive Filter Determination

The program uses the function *bos*, written by Blanchini and Sznaier. It requires as input

- the vectors m, n, d (they must be column vectors)
- the disturbance bounds β and γ
- the order of the filter
- a tentative value for $\mu > 0$

The following quantities are the software outputs

- the vectors a, b, c, that contain the coefficients of the filter denominator, of the filter numerator and of the polynomial that satisfies the constraint $m(\lambda)a(\lambda) - b(\lambda)n(\lambda) = c(\lambda)d(\lambda)$, respectively
- a margin value which should be positive and small (if it is not, another iteration is needed).

The polynomial whose coefficients are contained in c forms the numerator of the second part of the filter in presence of a known input, as explained in Chapter 3; the denominator is always a.

5.2.1 Function usage

At first the function is run for a tentative value of $\mu > 0$; if the margin is positive, μ must be decreased, else if the margin is negative, μ must be increased. The procedure must be repeated until μ is positive and as small as possible. The user should always verify that the value `check_eq_zero`, shown when the function is executed, is about zero, which results from a numerical check explained later. Besides, the code must be run until the margin `mrgn` is positive and as small as possible, to have the correct solution.

5.2.2 How does it work?

The code of the function is reported in Appendix A. Here we just analyse the basic principles.

The purpose is to express the optimization problem and to solve it by using `CVX`, a package for specifying and solving convex programs (see [7]). For further deepening, see some hints about convex optimization mentioned in Appendix B or consult the more exhaustive [8].

Let us remember the condition (2.18): $\mu\|[a_1\ a_2 \dots a_r]\|_1 + \|[c_0\ c_1 \dots c_r]\|_1 + \|b_0\ \dots b_r\|_1 \leq \mu$. The constant D has been assumed equal to 1, since its value can be reabsorbed in β. We need to find the value μ_{opt} for which equality holds when the vectors a, b, c satisfy the constraint $ma - bn = cd$.

First of all, the matrices M, N, D are created as convolution matrices from vectors m, n, d divided respectively by μ, β, γ. If c is a column vector and x is a column vector of

length l, then `convmtx(c,l)*x` is the same as `conv(c,x)`, so it corresponds to multiplying the polynomials represented by the two column vectors.

Then Φ is the partitioned matrix composed by M, $-N$, $-D$. The first column of Φ is multiplied by μ to restore its proper value (it had been previously divided by μ, but it should have not) and its sign is changed: it is renamed q. The remaining columns of Φ are renamed P. It is now possible to state the convex optimization problem in `CVX` environment. A variable x is defined as the concatenation of the coefficients $a_i\,\mu$ (except for $a_0 = 1$, which is already contained in q), $b_i\,\beta$ and $c_i\,\gamma$. The problem consists in minimizing the 1–norm of x subject to the constraint $Px = q$. The software `CVX` solves it numerically.

After the optimization has been carried out, the vectors a, b, c are extracted from x.

The margin is calculated as a difference between the tentative value μ and the 1–norm of x: the optimal value μ_{opt} should be approximately equal to the 1–norm of x, so we want the margin to be as small as possible (but positive).

The value `check_eq_zero` is calculated as the norm of $ma - bn - cd$ and it must be about zero to verify the validity of the constraint.

When the iteration procedure is over — the margin is positive and small enough — the latest μ value is taken as μ_{opt} and the latest vectors furnished by the program contain

- a the coefficients of the filter denominator
- b the coefficients of the filter numerator
- c the polynomial that satisfies the constraint $ma - bn = cd$. It is used as numerator of the second component of the filter when also a known input is present.

Hence the filter has been synthesized and it can be tested by simulation.

5.3 Simulation

Before the simulation, Bode diagrams of $b(\lambda)/a(\lambda)$ and, if in presence of a known input, $c(\lambda)/a(\lambda)$ are generated. The behaviour of the process and of the filter is reproduced to show the graph of z and $\hat{z}$: if the filter is good, they must be close to each other. Then the graph of the estimation error is displayed. After a transient period, the estimation error must be comprised between μ_{opt} and $-\mu_{opt}$ (its absolute value mustn't be greater than μ_{opt}); otherwise, the filter is not working properly. This fact is true up to numerical problems, as explained in Chapter 3.

5.3.1 Systems State Space Realization

From the polynomials forming the transfer functions of the process and of the filter, the program derives the state space realization of the two systems (the process and the filter).

The state of the two systems and the output are initialized to zero.

The state space realization is used during the simulation to update at every step the values of the variables.

5.3.2 Disturbance Generation

The disturbances v and w are randomly generated and they must respect the constraints

$$|v(t)| \leq \gamma,\ |w(t)| \leq \beta.$$

In one of the reported examples, we have tested the filter with two different kinds of noise: linear and sinusoidal. The former has a straight proceeding, but the angular factor can vary at each step of the simulation (the probability of variation is set by the user, as well as the minimum and maximum value that the angular factor can assume). The latter is a cosine of amplitude β or γ [1], whose pulsation can randomly change with a probability set by the user, within a prescribed range. In the other examples, only sinusoidal noise has been considered.

[1] so that the constraint is always respected

Piecewise Affine Functions

The angular coefficient α is set at a value comprised in the interval $(\alpha_{min}, \alpha_{max})$ by producing a random value with the MATLAB function `rand(N)`, that generates an N-by-N matrix with random entries, chosen from a uniform distribution on the interval (0.0,1.0). Then

$$\alpha = \alpha_{min} r + \alpha_{max}(1 - r), \text{ where r=rand(1)}$$

and the value of the noise at the time–instant k is

$$v(k) = v(k-1) + \alpha T_s, \text{ where } T_s \text{ is the sampling time.}$$

A saturation to $\pm\beta$ or $\pm\gamma$ is imposed to avoid the noise exceeding its limit. An example is shown in Fig. 5.1.

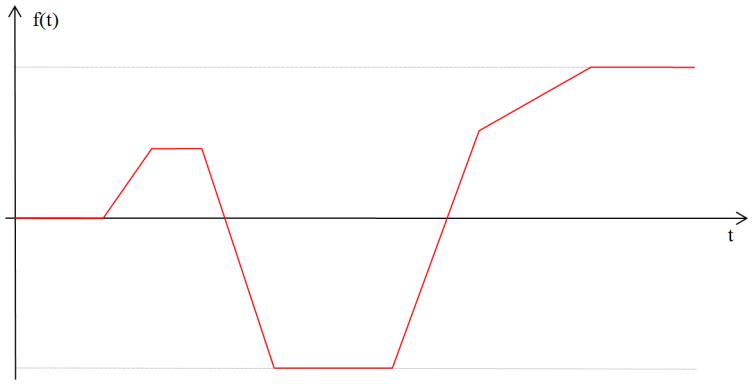


Figure 5.1: Piecewise affine input function

Sinusoidal Functions

The noise is generated as

$$v = \gamma \cos(\texttt{omega\_v}kT_s), \; w = \beta \cos(\texttt{omega\_w}kT_s),$$

where `omega_v=rand*omega_max`, `omega_w=rand*omega_max` and `omega_max` is the maximum pulsation value set by the user. An example is shown in Fig. 5.2.

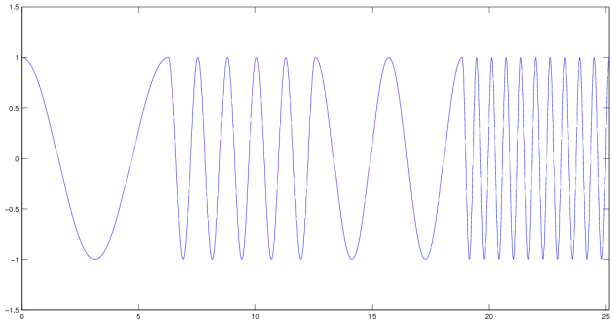


Figure 5.2: Sinusoidal input function

5.3.3 Comparison

A comparison between the filter obtained with the procedure described in [1] and the discrete–time Kalman filter has been performed, with sinusoidal disturbance functions.

The Bode diagrams of the two different filters are compared in a graph; another two graphs show z versus $\hat{z}_{eq}$ versus $\hat{z}_{kalman}$ and the estimation error $e_{eq} = z - \hat{z}_{eq}$ versus $e_{kalman} = z - \hat{z}_{kalman}$.

5.4 Some Illustrative Examples

Some examples are described next. Several examples will be considered in the following chapters without details.

5.4.1 Velocity Estimator

Consider the system

$$\begin{aligned}
\begin{bmatrix} \dot{x_1}(t) \\ \dot{x_2}(t) \end{bmatrix} &= \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} v(t) \\
y(t) &= \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + w(t) \\
z(t) &= \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix}
\end{aligned}$$

which is the space state realization of the mechanical system $m\ddot{q} = F$, with $x_1 = q$, $x_2 = \dot{q}$, $v = F/m$.

We have $y = q + w$ (the measured output is position, with its measure uncertainty) and $z = \dot{q}$ (the output to be estimated is the velocity).

Then

$$\begin{aligned}
&F = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix},\ G = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \\
&H_y = \begin{bmatrix} 1 & 0 \end{bmatrix},\ H_z = \begin{bmatrix} 0 & 1 \end{bmatrix}.
\end{aligned}$$

The discrete–time state space realization with sampling time T has matrices

$$\begin{aligned}
A &= e^{FT} = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} \\
B &= \int_0^T e^{F\xi} d\xi\ G = \begin{bmatrix} T^2/2 \\ T \end{bmatrix} \\
C_y &= H_y = \begin{bmatrix} 1 & 0 \end{bmatrix} \\
C_z &= H_z = \begin{bmatrix} 0 & 1 \end{bmatrix}
\end{aligned}$$

If we put $T = 1$, we have

$$\begin{aligned}
A &= \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \\
B &= \begin{bmatrix} 0.5 \\ 1 \end{bmatrix}
\end{aligned}$$

The transfer functions from v to z and from v to y are

$$\begin{aligned}
T_{v,z} &= C_z(\zeta I - A)^{-1}B = \frac{\zeta - 1}{(\zeta - 1)^2} \\
T_{v,y} &= C_y(\zeta I - A)^{-1}B = \frac{0.5\zeta + 0.5}{(\zeta - 1)^2}
\end{aligned}$$

where ζ is the variable of the zeta–transform, while the λ–transform, achieved by assuming $\zeta = 1/\lambda$, is

$$T_{v,z} = \frac{M(\lambda)}{d(\lambda)} = \frac{\lambda - \lambda^2}{1 - 2\lambda + \lambda^2}$$
$$T_{v,y} = \frac{N(\lambda)}{d(\lambda)} = \frac{0.5\lambda + 0.5\lambda^2}{1 - 2\lambda + \lambda^2}$$

so that

$$\begin{aligned} m(\lambda) &= \lambda - \lambda^2 \\ n(\lambda) &= 0.5\lambda + 0.5\lambda^2 \\ d(\lambda) &= 1 - 2\lambda + \lambda^2, \end{aligned}$$

and the input vectors for the function *bos* are

```
m=[0 1 -1]';
n=[1 0.5 0.5]';
d=[1 -2 1]';
```

We look for a filter of order $r = 3$, i.e. `ord_com = 3`, and we put $\beta = 1$, $\gamma = 1$. We start by assuming $\mu = 5$ and after some iterations we find the value $\mu_{opt} \approx 1.3$.

The resulting filter is

$$\frac{b(\lambda)}{a(\lambda)} = \frac{0.2500 - 0.2500\lambda^2}{1}$$

with

$$c(\lambda) = -0.2500 + 0.3750\lambda + 0.1250\lambda^2.$$

5.4.2 Equalized Performance Function $\mu(\gamma)$

The following example is derived from [1]. In the second order plant

$$\frac{M(\lambda)}{d(\lambda)} = \frac{\lambda^2}{1}, \ \frac{N(\lambda)}{d(\lambda)} = \frac{(1 - 0.5\lambda)(1 - 2\lambda)}{1}$$

the process and measurement noise satisfy $|v_k| \leq \gamma$ and $|w_k| \leq \beta$, respectively.

Since

$$\begin{aligned} m(\lambda) &= \lambda^2 \\ n(\lambda) &= (1 - 0.5\lambda)(1 - 2\lambda) \\ d(\lambda) &= 1, \end{aligned}$$

we have the following input vectors for the function *bos*:

```
m=[0 0 1]';
n=conv([1 -0.5],[1 -2])';
d=[1 0 0]';
```

We look for a filter of order $r = 3$, so that `ord_com = 3` and we put $\beta = 1$. Then we analyse the filter as a function of γ. We start by assuming $\mu = 3$, then after some iterations the right value of μ_{opt} will be reached.

For $0 < \gamma < 2$ the optimal equalized estimation is $\hat{z} = 0$ (zero filter: $b \approx 0$), with $\mu_{opt} \approx \gamma$ ([2]).

For instance:

[2] μ_{opt} is always slightly greater than γ

- $\gamma = 0.8$: $\mu_{opt} \approx 0.8$,

$$a = 1,$$

$$b = 1.0e-10(0.0074 - 0.1429\lambda + 0.0100\lambda^2 + 0.0047\lambda^3);$$

- $\gamma = 1$: $\mu_{opt} \approx 1$,

$$a = 1,$$

$$b = 1.0e-12(-0.1265 - 0.7065\lambda - 0.0968\lambda^2 - 0.0146\lambda^3);$$

- $\gamma = 1.5$: $\mu_{opt} \approx 1.5$,

$$a = 1,$$

$$b = 1.0e-9(0.0015 - 0.6652\lambda - 0.0081\lambda^2 - 0.0019\lambda^3);$$

- $\gamma = 1.95$: $\mu_{opt} \approx 1.95$,

$$a = 1,$$

$$b = 1.0e-9(-0.0094 - 0.1536\lambda - 0.0427\lambda^2 - 0.0107\lambda^3);$$

So for low values of γ ($0 < \gamma < 2$) the filter is zero.
If $\gamma = 2$, $\mu_{opt} \approx 2$. The problem admits multiple solutions, amongst them $\frac{b(\lambda)}{a(\lambda)} = 0$ and

$$\frac{b(\lambda)}{a(\lambda)} = \frac{-0.1505 - 0.4847\lambda - 0.1594\lambda^2 - 0.0460\lambda^3}{1 - 0.0661\lambda - 0.0445\lambda^2 - 0.0460\lambda^3},$$

with

$$c = 0.1505 + 0.1084\lambda + 0.0982\lambda^2 + 0.0661\lambda^3.$$

If $\gamma > 2$, the filter is

$$\frac{b(\lambda)}{a(\lambda)} = \frac{-0.2463 - 0.6158\lambda - 0.2933\lambda^2 - 0.1173\lambda^3}{1 - 0.1173\lambda^3}$$

with poles at 0.4895 and $-0.2448 \pm 0.4239j$. All the poles have equal modulus and, as expected, are stable.

$$c(\lambda) = 0.2463$$

The equalized performance level seems to be an affine linear function of γ:

- $\gamma = 2.1$: $\mu_{opt} \approx 2.03$
- $\gamma = 3$: $\mu_{opt} \approx 2.3$
- $\gamma = 4$: $\mu_{opt} \approx 2.6$
- $\gamma = 5$: $\mu_{opt} \approx 2.9$
- $\gamma = 8$: $\mu_{opt} \approx 3.7$
- $\gamma = 12$: $\mu_{opt} \approx 4.8$
- $\gamma = 22$: $\mu_{opt} \approx 7.6$
- $\gamma = 32$: $\mu_{opt} \approx 10.4$
- $\gamma = 102$: $\mu_{opt} \approx 30$,

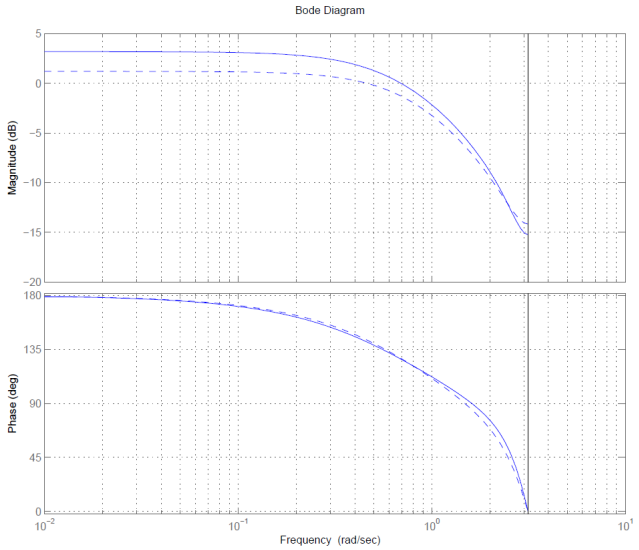


Figure 5.3: The equalized filter (plain) versus the $\mathcal{H}_2$ filter (dashed) frequency response (from [1])

while the filter is always the same.

The overall equalized performance level is given by the piecewise affine function of γ:

$$\mu_{opt}(\gamma) = \begin{cases} \gamma & \text{for} \quad 0 < \gamma < 2 \\ 2 + \kappa(\gamma - 2) & \text{for} \quad 2 < \gamma \end{cases}$$

with $\kappa \approx 0.28$.

In [1] the equalized filter is compared with the $\mathcal{H}_2$–optimal filter whose state space realization is

$$A_{h_2} = \begin{bmatrix} 0.3700 & 0.0750 \\ -0.2430 & 0.6076 \end{bmatrix} \quad B_{h_2} = \begin{bmatrix} -0.3700 \\ 0.2430 \end{bmatrix}$$

$$C_{h_2} = \begin{bmatrix} 1.1777 & -0.4443 \end{bmatrix} \quad D_{h_2} = [-0.1777] \,.$$

In Fig. 5.3 the frequency response of the equalized and the $\mathcal{H}_2$–optimal filters is reported and in Fig. 5.4 the simulations in the presence of random piecewise–constant noise are shown. The equalized filter has a slightly sharper cut–off effect.

The noise has been randomly generated by taking $v = \pm\gamma$ and $w = \pm\beta$ and randomly changing sign with probability $\pi = 1/10$ at each instant. In the reported simulation $\gamma = 10$ and $\beta = 1$. Several experiments with randomly generated sequences have shown that the worst case error produced by the equalized filter is always smaller than that produced by the $\mathcal{H}_2$ filter.

5.4.3 A Single Pole on the Stability Boundary

Now we consider a plant with one pole on the stability boundary.

$$\frac{M(\lambda)}{d(\lambda)} = \frac{\lambda^2}{1-\lambda}, \quad \frac{N(\lambda)}{d(\lambda)} = \frac{(1-0.5\lambda)(1-2\lambda)}{1-\lambda}$$

We have

$$\begin{aligned} m(\lambda) &= \lambda^2 \\ n(\lambda) &= (1-0.5\lambda)(1-2\lambda) \\ d(\lambda) &= 1-\lambda, \end{aligned}$$

so that the input vectors for the function *bos* are

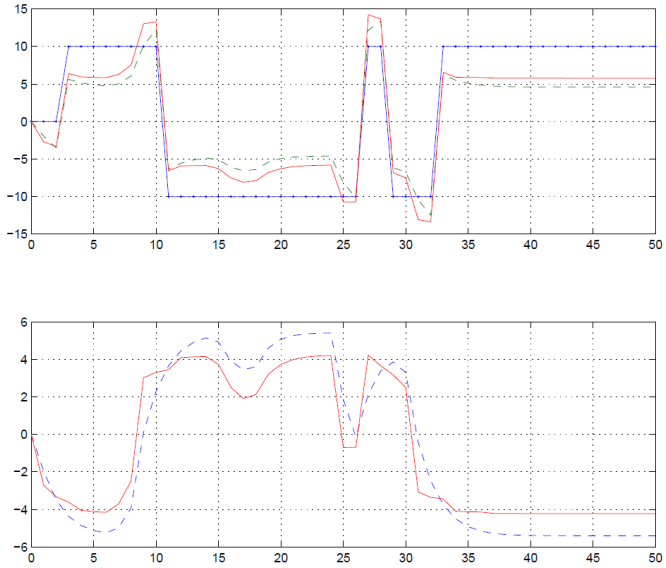


Figure 5.4: The equalized filter versus the $\mathcal{H}_2$ filter. Top figure: the reference (dotted), the $\mathcal{H}_2$ filter output (dashed), the equalized output (plain); bottom figure: the $\mathcal{H}_2$ filter error (dashed), the equalized filter error (plain) (from [1])

```
m=[0 0 1]';
n=conv([1 -0.5],[1 -2])';
d=[1 -1 0]';
```

For $r = 3$, $\beta = 1$, we study the behaviour of the filter when γ varies from $\gamma = 0.01$ to $\gamma = 100$: μ varies from $\mu = 2.04$ to $\mu = 59.5$, but the filter is always

$$\frac{b(\lambda)}{a(\lambda)} = \frac{-0.4912 - 0.7368\lambda - 0.3509\lambda^2 - 0.1404\lambda^3}{1 - 0.1404\lambda^3},$$

with $c = 0.4912$.

Its poles are $-0.2599 \pm 0.4501j$ and 0.5197, all with the same absolute value.

5.4.4 Two Poles on the Stability Boundary

The following example is derived from [1]. We consider the case of a plant with poles on the stability boundary.

$$\frac{M(\lambda)}{d(\lambda)} = \frac{\lambda^2}{1-\lambda^2}, \quad \frac{N(\lambda)}{d(\lambda)} = \frac{(1-0.5\lambda)(1-2\lambda)}{1-\lambda^2}$$

$$\begin{aligned} m(\lambda) &= \lambda^2 \\ n(\lambda) &= (1-2\lambda)(1-\lambda/2) \\ d(\lambda) &= 1-\lambda^2, \end{aligned}$$

so the input vectors for the function *bos* are

```
m=[0 0 1]';
n=conv([1 -0.5],[1 -2])';
d=[1 0 -1]';
```

The equalized filter corresponding to $\beta = 1$, $\gamma = 8$ and $r = 3$ is given by

$$\frac{b(\lambda)}{a(\lambda)} = \frac{-0.3268 - 0.8171\lambda - 0.3891\lambda^2 - 0.1556\lambda^3}{1 - 0.1556\lambda^3},$$

with

$$c = 0.3268,$$

and achieves an equalized performance level $\mu_{opt} \approx 5.1$

The poles are $-0.2689 \pm 0.4658j$ and 0.5379, all with the same absolute value.

With $\beta = 1$, the filter is the same even when $\gamma = 0.01$, $\gamma = 0.1$, $\gamma = 1$, $\gamma = 10$ and $\gamma = 100$. So the filter seems to be independent from the value of γ; of course the equalized performance level increases with γ.

5.4.5 A Stable Plant

Consider now a stable plant.

$$\frac{M(\lambda)}{d(\lambda)} = \frac{\lambda^2}{1 - 0.8\lambda}, \quad \frac{N(\lambda)}{d(\lambda)} = \frac{(1 - 0.5\lambda)(1 - 2\lambda)}{1 - 0.8\lambda}$$

$$\begin{aligned} m(\lambda) &= \lambda^2 \\ n(\lambda) &= (1 - 2\lambda)(1 - \lambda/2) \\ d(\lambda) &= 1 - 0.8\lambda, \end{aligned}$$

so that

```
m=[0 0 1]';
n=conv([1 -0.5],[1 -2])';
d=[1 -0.8 0]';
```

We put $r = 3$ and $\beta = 1$. For $\gamma < 0.4$, we obtain the zero filter (the process is stable) and the equalized performance is

$$\mu_{opt} \approx 5\gamma$$

For $\gamma = 0.4$, $\mu_{opt} \approx 2$ and

$$\frac{b(\lambda)}{a(\lambda)} = \frac{-0.1428 - 0.3973\lambda - 0.1739\lambda^2 - 0.1449\lambda^3}{1 - 0.0976\lambda - 0.1056\lambda^2 - 0.1449\lambda^3}$$

whose poles are $-0.2663 \pm 0.3988j$ and 0.6301.

For $\gamma > 0.4$ the evaluated filter is

$$\frac{b(\lambda)}{a(\lambda)} = \frac{-0.4098 - 0.6966\lambda - 0.3317\lambda^2 - 0.1327\lambda^3}{1 - 0.1327\lambda^3}$$

whose poles are $-0.2550 \pm 0.4417j$ and 0.5101, all with the same absolute value. For $\gamma = 2$, $\mu_{opt} \approx 2.76$.

Here again $\mu_{opt}(\gamma)$ seems to be piecewise affine:

$$\mu_{opt}(\gamma) = \begin{cases} 5\gamma & \text{for} \quad 0 < \gamma < 0.4 \\ 2 + \kappa(\gamma - 0.4) & \text{for} \quad 0.4 < \gamma \end{cases}$$

with $\kappa \approx 0.48$.

5.4.6 Higher Order Compensator

For the previous plant, we tried to synthesize a compensator of order $r = 4$. For $\beta = 1$, $\gamma = 2$ we have $\mu_{opt} \approx 2.72$, as expected lower than the equalized performance level assured by the compensator of order $r = 3$. The filter is

$$\frac{b(\lambda)}{a(\lambda)} = \frac{-0.4149 - 0.7054\lambda - 0.3485\lambda^2 - 0.1660\lambda^3 - 0.0664\lambda^4}{1 - 0.0664\lambda^4}$$

Its poles are ± 0.5076 and $0.0000 \pm 0.5076j$.

Chapter 6

Linear Systems with Disturbances

In this chapter we show some possible applications of the equalized filter proposed in [1] to systems affected by bounded disturbances, in absence of known inputs.

6.1 Three Floors Building

6.1.1 Model Construction

We consider the simplified model of a three-floor-building under seismic action, as shown in Fig. 6.1.

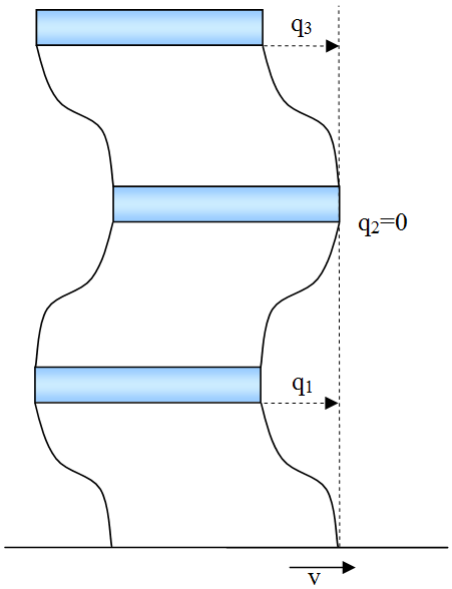


Figure 6.1: Three floors building under seismic action

Denoting by q_1, q_2, q_3 the displacements of each floor from the vertical, the masses m_1, m_2, m_3 are supposed to be concentrated in the floors. The system is governed by the equations

$$\begin{aligned} m_1\ddot{q_1}(t) &= -k_1 q_1(t) - k_{12}[q_1(t) - q_2(t)] - m_1 v(t) \\ m_2\ddot{q_2}(t) &= -k_{12}[q_2(t) - q_1(t)] - k_{23}[q_2(t) - q_3(t)] - m_2 v(t) \\ m_3\ddot{q_3}(t) &= -k_{23}[q_3(t) - q_2(t)] - m_3 v(t) \end{aligned}$$

where k_1, k_2, k_3 are the elastic constants. The terms $m_i v(t)$ represent the apparent forces originated by the ground acceleration, assumed as disturbance.

$$\begin{bmatrix} m_1 & 0 & 0 \\ 0 & m_2 & 0 \\ 0 & 0 & m_3 \end{bmatrix} \ddot{q}(t) = \begin{bmatrix} -(k_1+k_{12}) & k_{12} & 0 \\ k_{12} & -(k_{12}+k_{23}) & k_{23} \\ 0 & k_{23} & -k_{23} \end{bmatrix} q(t) + \begin{bmatrix} -m_1 \\ -m_2 \\ -m_3 \end{bmatrix} v(t)$$

The matricial equation written above is typical of a vibrant system, which in general is described as

$$M\ddot{\vartheta}(t) = -K\vartheta(t) + Ru(t)$$

where matrix M is always positive definite (in fact kinetic energy is $E_c = \frac{1}{2}\dot{\vartheta}^T(t)M\dot{\vartheta}(t) > 0$ if the system is moving) and matrix K is positive semidefinite (since potential energy $E_P = \frac{1}{2}\vartheta^T(t)K\vartheta(t) \geq 0$). In the considered case, also matrix K is positive definite.

To obtain a state space representation, the equation must be pre–multiplied by M^{-1}.

$$\ddot{q}(t) = -M^{-1}Kq(t) + M^{-1}Ru(t)$$

Assuming displacements and velocities as state variables, $x(t) = [q(t)\ \dot{q}(t)]^T$, we have

$$\frac{d}{dt}\begin{bmatrix} q(t) \\ \dot{q}(t) \end{bmatrix} = \begin{bmatrix} 0 & I \\ -M^{-1}K & 0 \end{bmatrix}\begin{bmatrix} q(t) \\ \dot{q}(t) \end{bmatrix} + \begin{bmatrix} 0 \\ M^{-1}R \end{bmatrix} u(t)$$

Take $S = M^{-1}K$, then matrix A can be written as

$$A = \begin{bmatrix} 0 & I \\ -S & 0 \end{bmatrix}$$

Now we assume $m_1 = m_2 = m_3 = 1$; if we choose as output to be estimated the shift of the first floor, $z = q_1$, and as measured output the shift of the second one, $y = q_2 + w$, we obtain

$$\begin{aligned} \dot{x} &= Ax + Bv \\ z &= C_z x \\ y &= C_y x + w, \end{aligned}$$

where

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ -(k_1+k_{12}) & k_{12} & 0 & 0 & 0 & 0 \\ k_{12} & -(k_{12}+k_{23}) & k_{23} & 0 & 0 & 0 \\ 0 & k_{23} & -k_{23} & 0 & 0 & 0 \end{bmatrix},\ B = \begin{bmatrix} 0 \\ 0 \\ 0 \\ -1 \\ -1 \\ -1 \end{bmatrix},$$

$$C_z = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix},\ C_y = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

If we consider also friction among the floors, matrix A becomes

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ -(k_1+k_{12}) & k_{12} & 0 & -\varphi & 0 & 0 \\ k_{12} & -(k_{12}+k_{23}) & k_{23} & 0 & -\varphi & 0 \\ 0 & k_{23} & -k_{23} & 0 & 0 & -\varphi \end{bmatrix}$$

where φ is a small positive number that expresses the friction coefficient.

6.1.2 Synthesis and Simulation

If we put $k_1 = 2$, $k_{12} = k_{23} = 1$, $\varphi = 0.01$, matrix A is

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ -3 & 1 & 0 & -0.01 & 0 & 0 \\ 1 & -2 & 1 & 0 & -0.01 & 0 \\ 0 & 1 & -1 & 0 & 0 & -0.01 \end{bmatrix}$$

The eigenvalues of matrix A are

$$\begin{aligned} &-0.0050 + 1.9318j \\ &-0.0050 - 1.9318j \\ &-0.0050 + 1.4142j \\ &-0.0050 - 1.4142j \\ &-0.0050 + 0.5176j \\ &-0.0050 - 0.5176j \end{aligned}$$

The matrix obtained by discretization is

$$\begin{bmatrix} -0.1254 & 0.3179 & 0.0338 & 0.5739 & 0.1283 & 0.0072 \\ 0.3179 & 0.2263 & 0.3856 & 0.1283 & 0.7093 & 0.1427 \\ 0.0338 & 0.3856 & 0.5781 & 0.0072 & 0.1427 & 0.8448 \\ -1.5933 & 0.3244 & 0.1211 & -0.1312 & 0.3166 & 0.0338 \\ 0.3244 & -1.1477 & 0.5667 & 0.3166 & 0.2192 & 0.3842 \\ 0.1211 & 0.5667 & -0.7022 & 0.0338 & 0.3842 & 0.5696 \end{bmatrix}$$

Its eigenvalues are

$$\begin{aligned} &0.8647 + 0.4923j \\ &0.8647 - 0.4923j \\ &-0.3515 + 0.9309j \\ &-0.3515 - 0.9309j \\ &0.1552 + 0.9828j \\ &0.1552 - 0.9828j \end{aligned}$$

Since all the eigenvalues of matrix A have negative real part and all the eigenvalues of the matrix obtained by discretization have absolute value smaller than 1 (0.995), the system is stable.

With sampling time $T_s = 1$, $\beta = 1$, $\gamma = 10$, $r = 7$, we obtain $\mu_{opt} \approx 14.45$ and the filter is a *Finite Impulse Response* (FIR)

$$\frac{b(\lambda)}{a(\lambda)} \approx \frac{0.6744 - 0.5856\lambda + 0.7321\lambda^2 - 0.6186\lambda^3 + 0.8429\lambda^5 - 0.8173\lambda^6 + 0.2942\lambda^7}{1}(^1),$$

with

$$c \approx 0 - 0.0888\lambda - 0.0294\lambda^2 + 0.3143\lambda^3 + 0.1931\lambda^4 - 0.2093\lambda^5 + 0.1467\lambda^7.$$

The Bode diagram of the filter is shown in Fig. 6.2, while Fig. 6.3 shows the result of some simulations obtained with sinusoidal input functions and parameters

```
omega_max = 5;
p_change_v = 0.8;
p_change_w = 0.8;
```

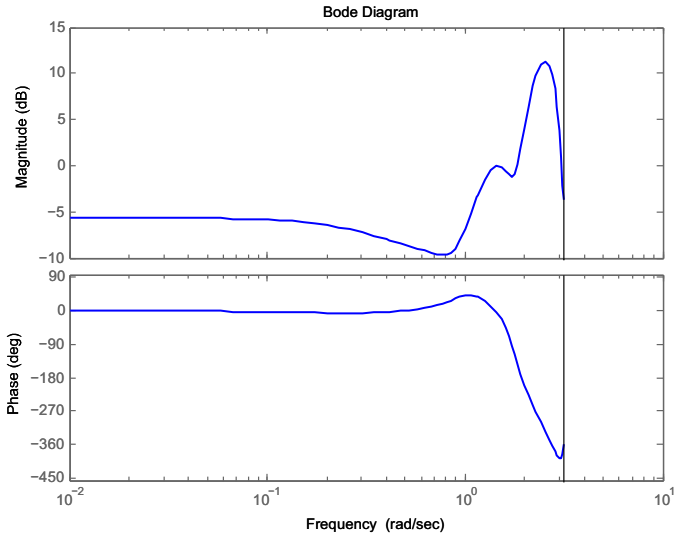


Figure 6.2: Bode diagram of the filter for the building

The behaviour of the filter is good: $\hat{z}$ follows accurately z.

As expected from an equalized filter obtained with the procedure explained in [1], the estimation error is never greater in norm than the optimal equalized filtering level μ_{opt}.

6.1.3 Comparison with Kalman Filter

The discrete–time Kalman filter transfer function is

$$\frac{b_{kalman}(\lambda)}{a_{kalman}(\lambda)} \approx \frac{0.3059\lambda - 0.1255\lambda^2 + 0.4119\lambda^3 - 0.2165\lambda^4 + 0.2416\lambda^5 - 0.2806\lambda^6}{1 - 0.4318\lambda + 1.5683\lambda^2 - 0.7715\lambda^3 + 1.0658\lambda^4 - 0.5019\lambda^5 + 0.2500\lambda^6}$$

and its poles are $-0.3524 \pm 0.8875j$, $0.1904 \pm 0.8995j$, $0.3779 \pm 0.4261j$.

We compare this Kalman filter with the r–equalized filter with $r = 6$, which attains $\mu_{opt} \approx 15.09$ and is

$$\frac{b(\lambda)}{a(\lambda)} \approx \frac{0.7577 - 0.9996\lambda + 1.3084\lambda^2 - 1.4336\lambda^3 + 0.8578\lambda^4 + 0.1127\lambda^5 - 0.3403\lambda^6}{1} \quad (^2)$$

with

$$c \approx 0 - 0.0475\lambda - 0.1212\lambda^2 + 0.1728\lambda^3 + 0.0000007\lambda^4 - 0.4130\lambda^5 - 0.1697\lambda^6.$$

Its poles are ± 0.0667, $0.0334 \pm 0.0578j$, $-0.0334 \pm 0.0578j$. They are approximately zero, namely the filter is an FIR.

The Bode diagram of the Kalman filter is compared with that of the r–equalized filter in Fig. 6.4.

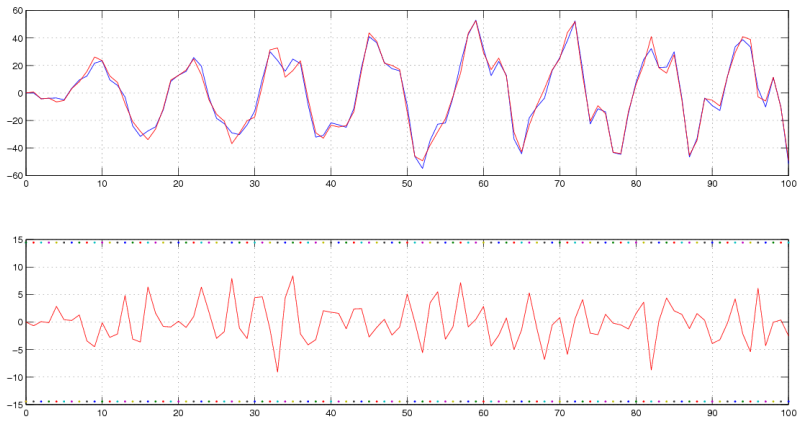
The result of some comparative simulations obtained with sinusoidal input functions and parameters

```
omega_max = 5;
p_change_v = 0.2;
p_change_w = 0.2;
```

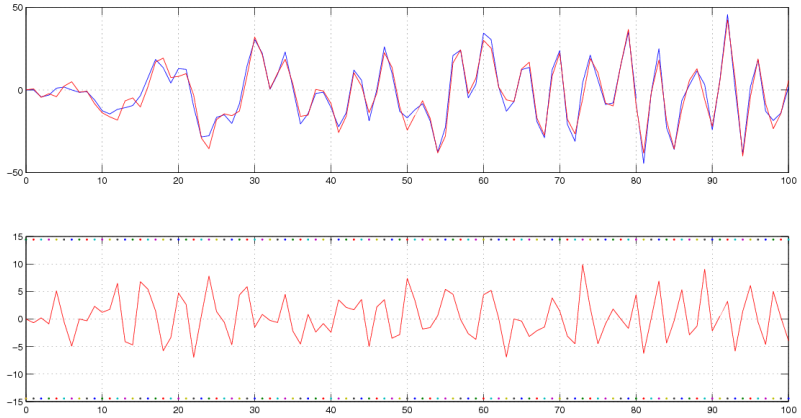
is shown in Fig. 6.5.

[1] The values a_i with $i > 0$ are not exactly zero, but they are very close to zero

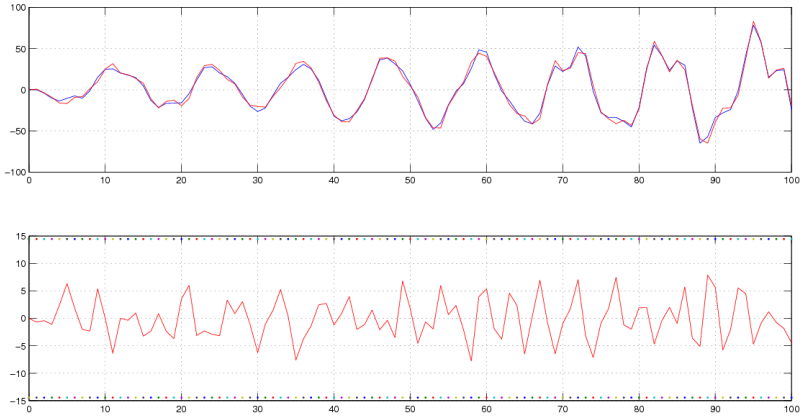
[2] The values a_i with $i > 0$ are not exactly zero, but they are very close to zero



(a)



(b)



(c)

Figure 6.3: Simulations of the filter behaviour for the building under seismic action with sinusoidal input function. Above z, in blue, is compared with $\hat{z}$, in red; below, the estimation error is shown and its limits $\pm\mu_{opt}$ are marked

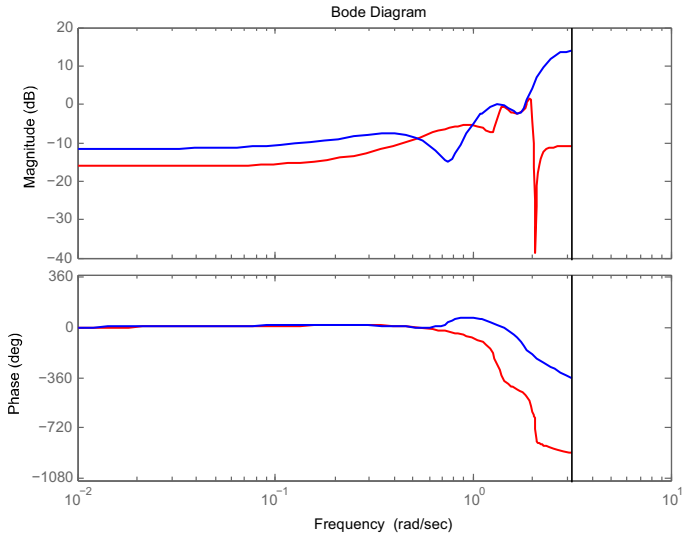


Figure 6.4: Bode diagram of the Kalman filter (in red) and of the equalized filter (in blue) for the building

The frequency response of the equalized filter is smoother. Both the filters work well, but several experiments with randomly generated noise reveal that the worst case estimation error of the equalized filter is always smaller, with an often noticeable improvement.

6.2 System Stability and Filter Behaviour

In this section we aim to analyse the link between the stability of the system considered and the effectiveness of the filter synthesized.

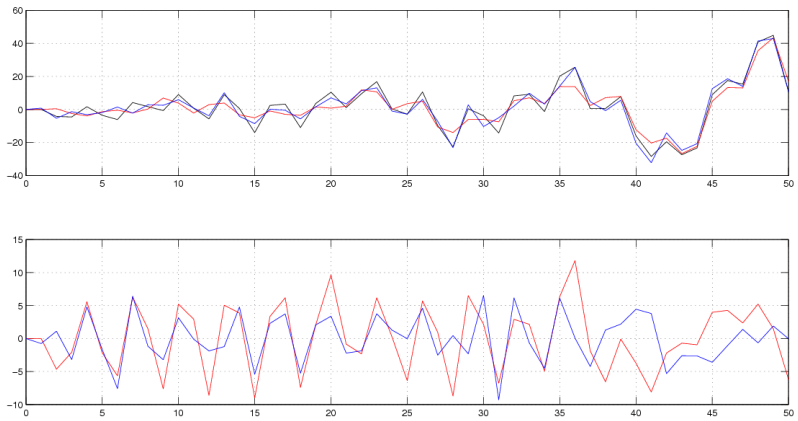
The system we previously considered is stable and the equalized filter works very well. Consider now the system

$$\begin{aligned} \dot{x} &= Fx + Gv \\ z &= H_z x \\ y &= H_y x + w, \end{aligned}$$

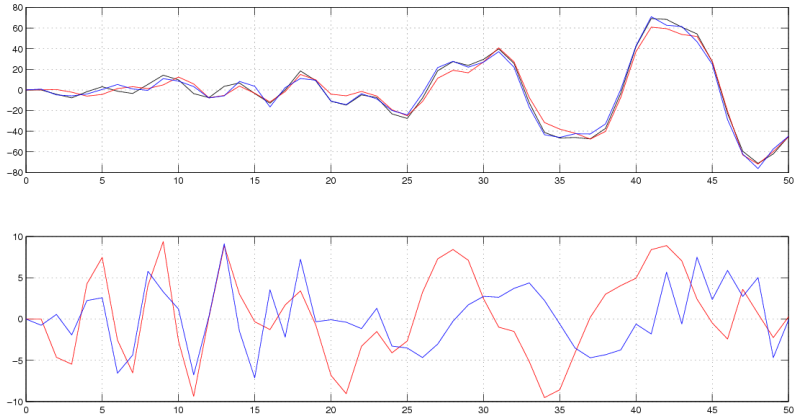
with

$$F = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ -k_1 & 1 & 0 & 0 & 0 & 0 \\ 1 & -k_2 & 1 & 0 & 0 & 0 \\ 0 & 1 & -k_3 & 0 & 0 & 0 \end{bmatrix}, \; G = \begin{bmatrix} 0 \\ 0 \\ 0 \\ -1 \\ -1 \\ -1 \end{bmatrix},$$
$$H_z = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \; H_y = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

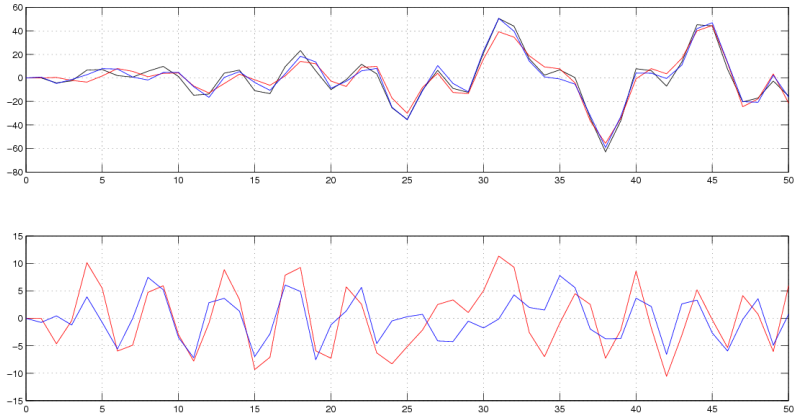
If we vary the values of k_1, k_2, k_3, we can obtain either a marginally stable system or an unstable system. We are interested in the behaviour of the filter in these different cases.



(a)



(b)



(c)

Figure 6.5: Simulations for the building under seismic action with sinusoidal input function. Above z, in black, is compared with $\hat{z}_{eq}$, in blue, and $\hat{z}_{kalman}$, in red; below, the estimation error e_{eq}, in blue, is compared with e_{kalman}, in red

6.2.1 Marginally Stable Oscillator

If $k_1 = 1$, $k_2 = 2$, $k_3 = 1$, F, matrix of the continuous–time system, is

$$F = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ -1 & 1 & 0 & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 \end{bmatrix}$$

and all its eigenvalues have real part equal to 0:

$$\begin{aligned} 0.0000 + 1.7321j \\ 0.0000 - 1.7321j \\ 0.0000 + 1.0000j \\ 0.0000 - 1.0000j \\ 0.0000 \\ -0.0000 \end{aligned}$$

The matrix A obtained by discretization of the previous system is

$$A = \begin{bmatrix} 0.5767 & 0.3869 & 0.0364 & 0.8490 & 0.1434 & 0.0076 \\ 0.3869 & 0.2263 & 0.3869 & 0.1434 & 0.7132 & 0.1434 \\ 0.0364 & 0.3869 & 0.5767 & 0.0076 & 0.1434 & 0.8490 \\ -0.7057 & 0.5699 & 0.1358 & 0.5767 & 0.3869 & 0.0364 \\ 0.5699 & -1.1397 & 0.5699 & 0.3869 & 0.2263 & 0.3869 \\ 0.1358 & 0.5699 & -0.7057 & 0.0364 & 0.3869 & 0.5767 \end{bmatrix}$$

Its eigenvalues are

$$\begin{aligned} -0.1606 + 0.9870j \\ -0.1606 - 0.9870j \\ 0.5403 + 0.8415j \\ 0.5403 - 0.8415j \\ 1.0000 + 0.0000j \\ 1.0000 - 0.0000j \end{aligned}$$

and all have absolute value equal to 1, as expected. So the system is marginally stable.

The filter obtained with parameters $T_s = 1$, $\beta = 1$, $\gamma = 10$, $r = 7$ has equalized performance $\mu_{opt} \approx 1.1$ and is a simple constant

$$\frac{b(\lambda)}{a(\lambda)} \approx \frac{0.9996}{1}([3]),$$

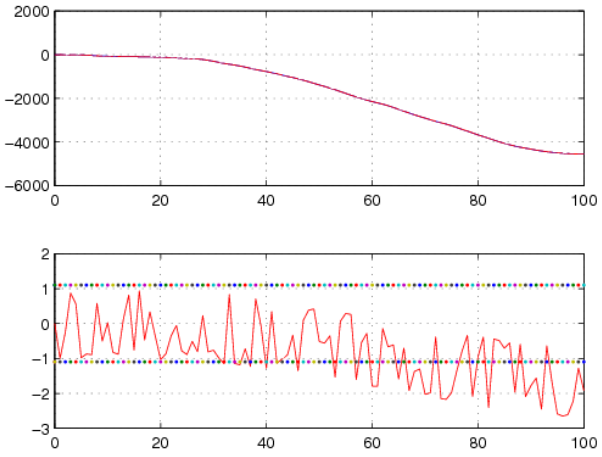
with

$$c \approx 0.$$

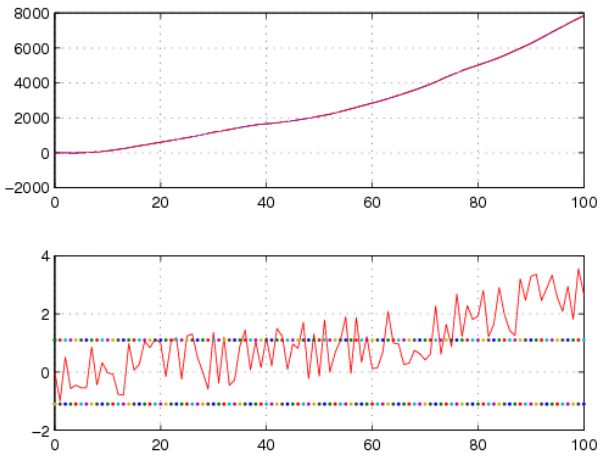
We show in Fig. 6.6 the behaviour of the filter with simulation parameters

```
omega_max = 5;
p_change_v = 0.8;
p_change_w = 0.8;
```

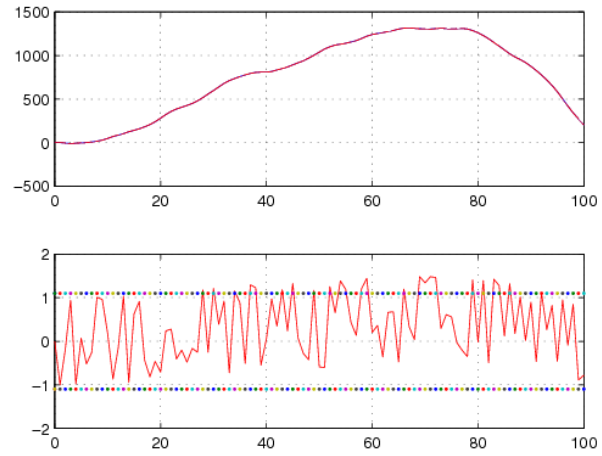
[3] The values a_i and b_i with $i > 0$ are not exactly zero, but they are very close to zero



(a)



(b)



(c)

Figure 6.6: Simulations of the filter behaviour for the first marginally stable system with sinusoidal input function. Above z, in blue, is compared with $\hat{z}$, in red; below, the estimation error is shown and its limits $\pm\mu_{opt}$ are marked

Consider another example.

If $k_1 = 2$, $k_2 = 2$, $k_3 = 2$, F, matrix of the continuous–time system, is

$$F = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ -2 & 1 & 0 & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 & 0 \\ 0 & 1 & -2 & 0 & 0 & 0 \end{bmatrix}$$

and all its eigenvalues have real part equal to 0:

$$\begin{aligned} -0.0000 + 1.8478j \\ -0.0000 - 1.8478j \\ 0.0000 + 1.4142j \\ 0.0000 - 1.4142j \\ 0.0000 + 0.7654j \\ 0.0000 - 0.7654j \end{aligned}$$

The matrix A obtained by discretization of the previous system is

$$A = \begin{bmatrix} 0.1899 & 0.3516 & 0.0340 & 0.7057 & 0.1360 & 0.0072 \\ 0.3516 & 0.2238 & 0.3516 & 0.1360 & 0.7129 & 0.1360 \\ 0.0340 & 0.3516 & 0.1899 & 0.0072 & 0.1360 & 0.7057 \\ -1.2754 & 0.4409 & 0.1216 & 0.1899 & 0.3516 & 0.0340 \\ 0.4409 & -1.1538 & 0.4409 & 0.3516 & 0.2238 & 0.3516 \\ 0.1216 & 0.4409 & -1.2754 & 0.0340 & 0.3516 & 0.1899 \end{bmatrix}$$

Its eigenvalues are

$$\begin{aligned} 0.7211 + 0.6928j \\ 0.7211 - 0.6928j \\ -0.2734 + 0.9619j \\ -0.2734 - 0.9619j \\ 0.1559 + 0.9878j \\ 0.1559 - 0.9878j \end{aligned}$$

and all have absolute value equal to 1.

So the system is marginally stable.

The filter obtained with parameters $T_s = 1$, $\beta = 1$, $\gamma = 10$, $r = 7$ has equalized performance $\mu_{opt} \approx 9.1$ and is an FIR

$$\frac{b(\lambda)}{a(\lambda)} \approx \frac{0.9296 - 0.7267\lambda + 1.0272\lambda^2 - 0.7569\lambda^3 + 0.5553\lambda^5 - 0.5592\lambda^6 + 0.2012\lambda^7}{1} (^4),$$

with

$$c \approx 0 + 0.0014\lambda - 0.1071\lambda^2 + 0.0361\lambda^4 - 0.1709\lambda^5 + 0.1005\lambda^7.$$

The Bode diagram of the filter is shown in Fig. 6.7.

The behaviour of the filter with simulation parameters

```
omega_max = 5;
p_change_v = 0.8;
p_change_w = 0.8;
```

is shown in Fig. 6.8.

[4] The values a_i with $i > 0$ are not exactly zero, but they are very close to zero

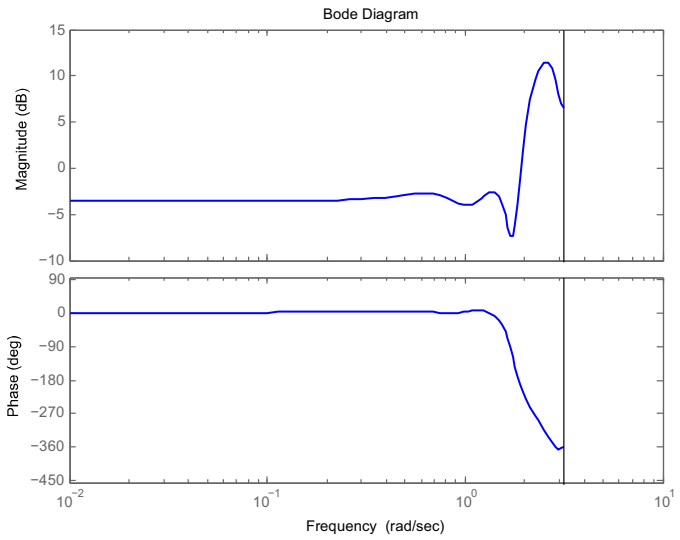


Figure 6.7: Bode diagram of the filter for the second marginally stable system

6.2.2 Unstable System

An unstable system can be obtained by adding a positive term $+k_{22}$ to the element $5, 2$ of matrix A of the previous example (three floors building under seismic action). Such kind of models are achieved in the presence of unstable parts in mechanical plants, e.g. an inverted pendulum connected to elastic components.

If $k_1 = 2$, $k_2 = 1$, $k_3 = 1$, F, matrix of the continuous–time system, is

$$F = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ -2 & 1 & 0 & 0 & 0 & 0 \\ 1 & -1 & 1 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 \end{bmatrix}$$

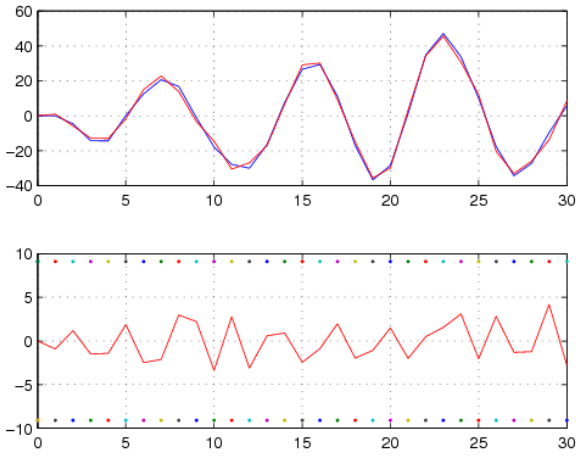
and its eigenvalues are:

$$\begin{aligned} 0.0000 + 1.6739j \\ 0.0000 - 1.6739j \\ 0.0000 + 1.2021j \\ 0.0000 - 1.2021j \\ 0.4970 \\ -0.4970 \end{aligned}$$

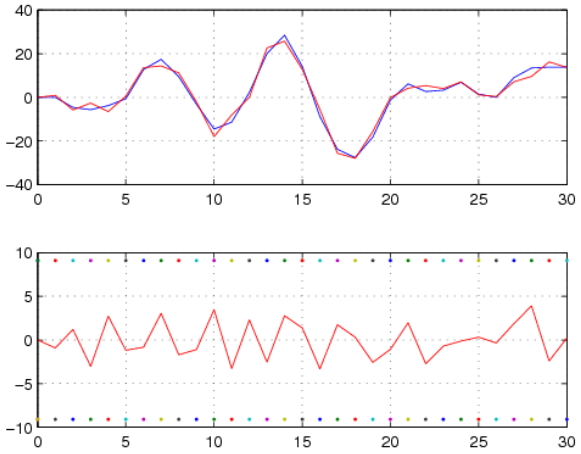
One of them has a positive real part.

We call A the matrix obtained by discretization of the previous system.

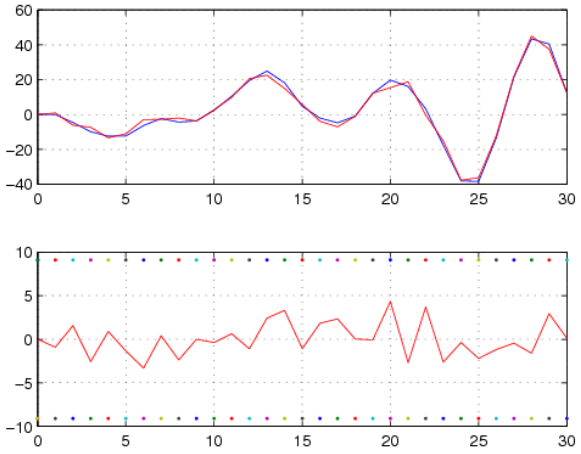
$$A = \begin{bmatrix} 0.1911 & 0.3869 & 0.0364 & 0.7058 & 0.1434 & 0.0076 \\ 0.3869 & 0.6144 & 0.4233 & 0.1434 & 0.8568 & 0.1510 \\ 0.0364 & 0.4233 & 0.5780 & 0.0076 & 0.1510 & 0.8492 \\ -1.2683 & 0.5700 & 0.1358 & 0.1911 & 0.3869 & 0.0364 \\ 0.5700 & -0.5625 & 0.7058 & 0.3869 & 0.6144 & 0.4233 \\ 0.1358 & 0.7058 & -0.6983 & 0.0364 & 0.4233 & 0.5780 \end{bmatrix}$$



(a)



(b)



(c)

Figure 6.8: Simulations of the filter behaviour for the second marginally stable system with sinusoidal input function. Above z, in blue, is compared with $\hat{z}$, in red; below, the estimation error is shown and its limits $\pm\mu_{opt}$ are marked

Its eigenvalues are

$$\begin{aligned} 1.6437 \\ -0.1029 + 0.9947j \\ -0.1029 - 0.9947j \\ 0.3604 + 0.9328j \\ 0.3604 - 0.9328j \\ 0.6084 \end{aligned}$$

and the first one has absolute value greater than 1. So the system is unstable. The filter obtained with parameters $T_s = 1$, $\beta = 1$, $\gamma = 10$, $r = 7$ has equalized performance $\mu_{opt} \approx 25.04$ and is an FIR

$$\frac{b(\lambda)}{a(\lambda)} \approx \frac{0.6922 - 0.5261\lambda + 0.1775\lambda^2 - 0.5744\lambda^4 + 0.8428\lambda^5 + 0.6256\lambda^6 - 0.5573\lambda^7}{1}(^5),$$

with

$$c \approx 0 - 0.0892\lambda + 0.2727\lambda^3 + 0.0730\lambda^4 - 0.5818\lambda^5 - 0.7716\lambda^6 - 0.3008\lambda^7.$$

The Bode diagram of the filter is shown in Fig. 6.9.

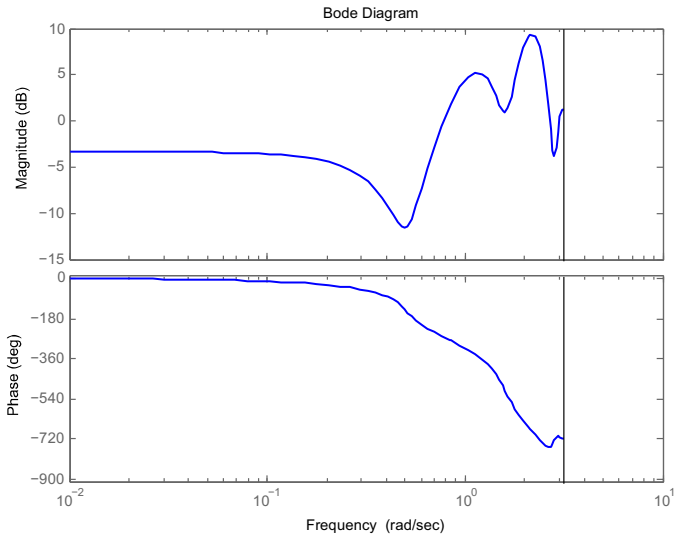


Figure 6.9: Bode diagram of the filter for unstable system

The behaviour of the filter with simulation parameters

```
omega_max = 5;
p_change_v = 0.8;
p_change_w = 0.8;
```

is shown in Fig. 6.10.

[5]The values a_i with $i > 0$ are not exactly zero, but they are very close to zero

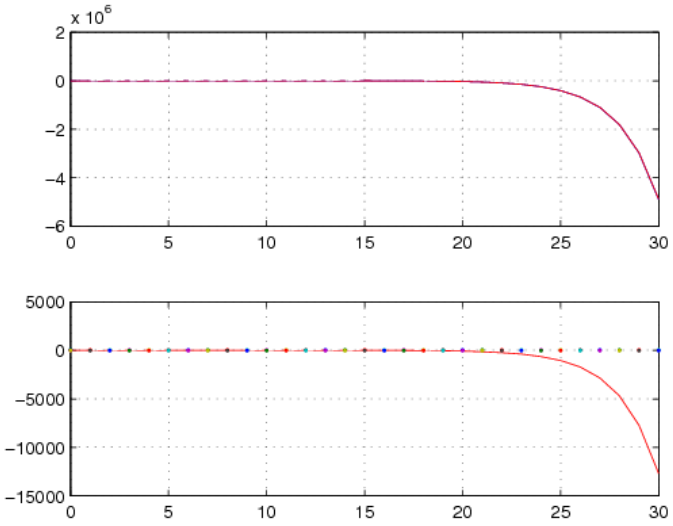


Figure 6.10: Simulation of the filter for unstable system with sinusoidal input function. Above z, in blue, is compared with $\hat{z}$, in red; below, the estimation error is shown and its limits $\pm\mu_{opt}$ are marked

6.2.3 Remarks

As the previous examples show, the filter works well when the system is stable, but this is not always the case when it is marginally stable or unstable.

When the system is unstable, the estimation error during simulations may exponentially diverge. When the system is only marginally stable, it can violate its bounds given by the optimal equalized filtering level μ_{opt}.

The reason of such a behaviour is due to numerical effects. In fact, as it is demonstrated in the proof of Lemma 3, in the transfer function that produces the estimation error, the modes of the system are cancelled and replaced by the modes of the filter (which are guaranteed to be stable). This is indeed a pole–zero cancellation, which can cause problems if the modes are not asymptotically stable. Since the cancellation is not perfect, the unstable or marginally stable modes can affect the behaviour of the estimation error anyway.

Indeed when an unstable mode is still present in the transfer function, due to a non–exact pole–zero cancellation, of course it leads the error to diverge exponentially. When a marginally stable mode is still present, if the error happens to stray from zero, then it is not necessarily brought back to zero, but it might go adrift.

The examples of this section show that it is dangerous to apply the filter to systems which are not stable, because an imperfect pole–zero cancellation (caused by numerical effects and rounding) can lead to instability. It is therefore recommended first of all to stabilise the system, maybe with a feedback control, and then to use the filter in order to estimate some quantities of the system.

6.3 A Car Suspension Model

6.3.1 Model Construction

We want to study a car suspension system, by means of the one–wheel–model loaded by a constant mass M. We denote by m the wheel mass, by K the elastic constant of the suspension, by k the elastic constant of the tyre and by H the damping coefficient of the shock absorber. q_2 and q_1 are the displacements of the two masses from the equilibrium position they assume when the car is motionless. The disturbance v is the vertical shift of the tread caused by the irregularities of the road. The model is drawn in Fig. 6.11.

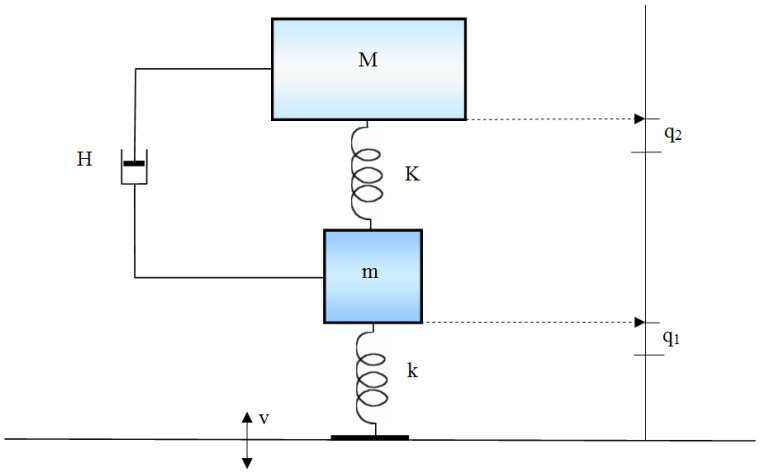


Figure 6.11: Car suspension

The equilibrium equations of the forces that act on masses M and m are

$$\begin{aligned} m\ddot{q_1}(t) &= -k[q_1(t) - v(t)] - K[q_1(t) - q_2(t)] - H[\dot{q_1}(t) - \dot{q_2}(t)] \\ M\ddot{q_2}(t) &= -K[q_2(t) - q_1(t)] - H[\dot{q_2}(t) - \dot{q_1}(t)] \end{aligned}$$

For convenience we take (although not very realistically) $m = M$ and $k/m = k_1$, $K/m = K/M = k_2$, $H/m = H/M = h$.

$$\begin{aligned} \ddot{q_1}(t) &= -k_1[q_1(t) - v(t)] - k_2[q_1(t) - q_2(t)] - h[\dot{q_1}(t) - \dot{q_2}(t)] \\ \ddot{q_2}(t) &= -k_2[q_2(t) - q_1(t)] - h[\dot{q_2}(t) - \dot{q_1}(t)] \end{aligned}$$

We assume that z is the shift of the wheel mass, which has to be estimated from y, the distance between the two masses.

$$\begin{aligned} z(t) &= q_1(t) \\ y(t) &= [q_1(t) - q_2(t)] + Dw(t) \end{aligned}$$

and we put D=1. If we assume as state variables position and velocity of the two masses, i.e. $x_1 = q_1$, $x_2 = q_2$, $x_3 = \dot{q_1}$, $x_4 = \dot{q_2}$, we obtain the equations

$$\begin{aligned} \dot{x} &= Ax + Bv \\ z &= C_z x \\ y &= C_y x + w, \end{aligned}$$

where

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -(k_1 + k_2) & k_2 & -h & h \\ k_2 & -k_2 & h & -h \end{bmatrix}, \; B = \begin{bmatrix} 0 \\ 0 \\ k_1 \\ 0 \end{bmatrix},$$

$$C_z = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}, \; C_y = \begin{bmatrix} 1 & -1 & 0 & 0 \end{bmatrix}.$$

6.3.2 Filter Synthesis

We choose the values $k_1 = 2$, $k_2 = 3$, $h = 0.1$ and the sampling time $T_s = 1$. If we put $\beta = 1$, $\gamma = 10$, $r = 6$, we obtain $\mu_{opt} \approx 18.7$ and the filter is an FIR

$$\frac{b(\lambda)}{a(\lambda)} \approx \frac{-0.3106 - 0.3808\lambda + 0.8183\lambda^2 + 1.1649\lambda^3 + 0.7725\lambda^4 + 0.4664\lambda^5 - 0.0000\lambda^6}{1} ([6]),$$

with

$$c \approx 0 + 0.7938\lambda + 0.4408\lambda^2 - 0.0000\lambda^3 - 0.0000\lambda^4 - 0.2357\lambda^5 - 0.0000\lambda^6.$$

The Bode diagram of the filter is shown in Fig. 6.12.

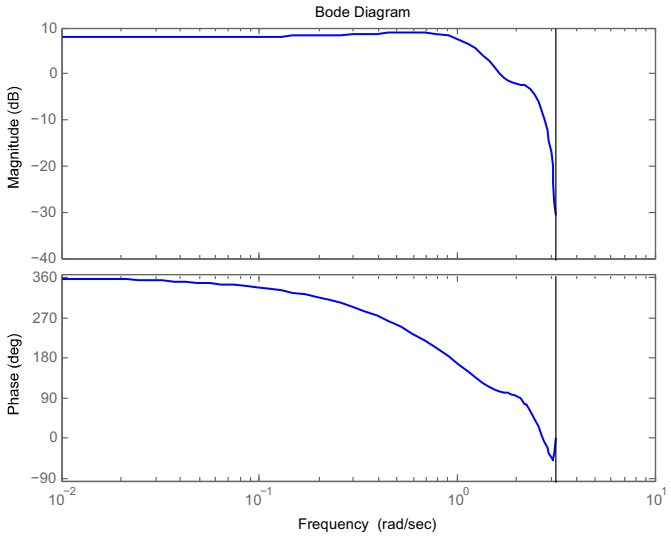


Figure 6.12: Bode diagram of the filter for car suspension, in the case $k_1 = 2$, $k_2 = 3$

If, instead, we choose $k_1 = 1$, $k_2 = 1$, with the other parameters unchanged, we obtain $\mu_{opt} \approx 17.3$ and the filter turns out to be not an FIR

$$\frac{b(\lambda)}{a(\lambda)} \approx \frac{-0.3950 + 0.0080\lambda - 0.1708\lambda^2 + 0.9199\lambda^4 - 0.0644\lambda^5 + 0.3857\lambda^6}{1 + 0.1115\lambda^6},$$

with

$$c \approx 0 + 0.5547\lambda + 0.6649\lambda^2 - 0.1097\lambda^6.$$

Its poles are $-0.6008 \pm 0.3469j$, $0.6008 \pm 0.3469j$, $-0.0000 \pm 0.6938j$.

The new Bode diagram of the filter is shown in Fig. 6.13.

6.3.3 Simulation

Sinusoidal Input Function

The result of some simulations obtained with parameters

```
omega_max = 5;
p_change_v = 0.2;
p_change_w = 0.5;
```

is shown in Fig. 6.14, in the case $k_1 = 2$, $k_2 = 3$, and in Fig. 6.15, in the case $k_1 = 1$, $k_2 = 1$.

[6]The values a_i with $i > 0$ are not exactly zero, but they are very close to zero

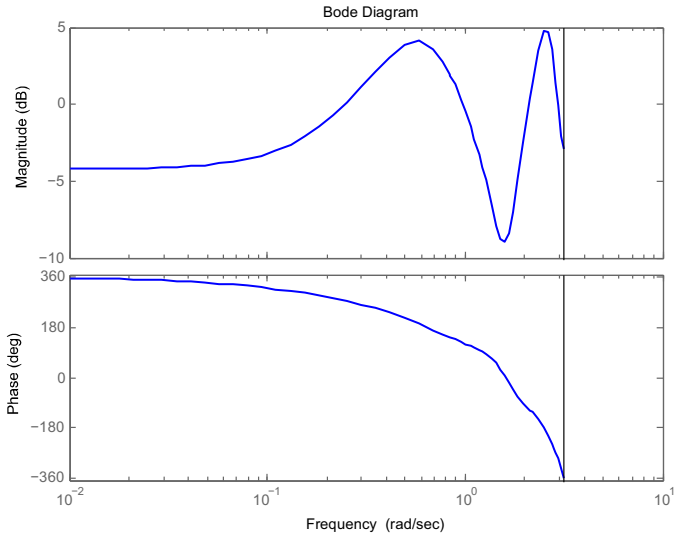


Figure 6.13: Bode diagram of the filter for car suspension, in the case $k_1 = 1$, $k_2 = 1$

Piecewise Affine Input Function

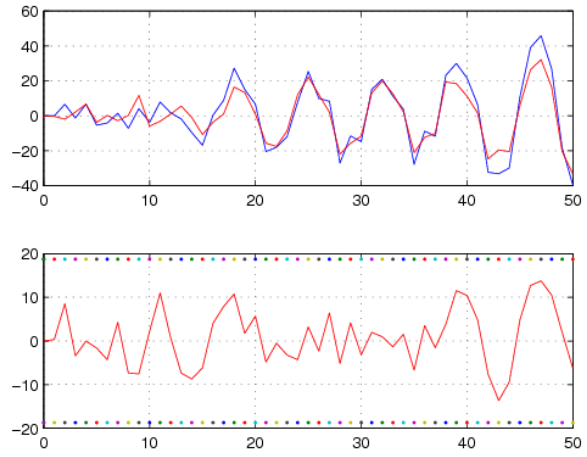
The result of some simulations obtained with parameters

```
alfa_min = -2;
alfa_max = 2;
ang_min = -1;
ang_max = 1;
p_change_v = 0.2;
p_change_w = 0.5;
```

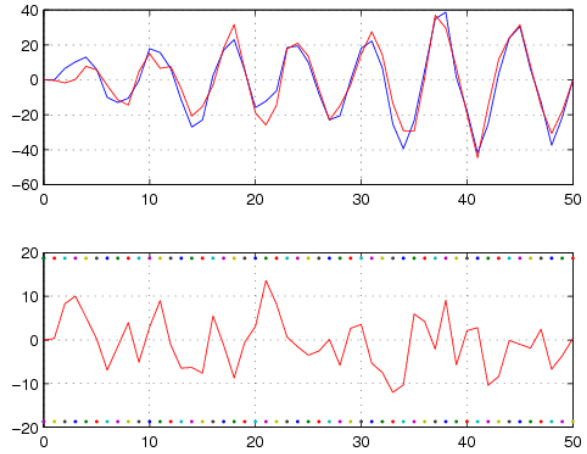
is shown in Fig. 6.16, in the case $k_1 = 2$, $k_2 = 3$, and in Fig. 6.17, in the case $k_1 = 1$, $k_2 = 1$.

Remarks

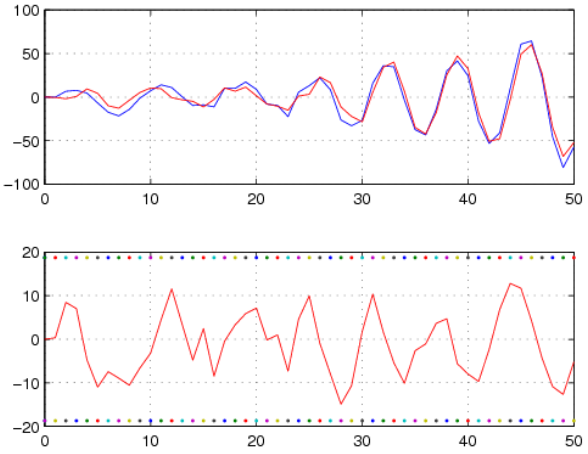
In both cases the filter works well, with sinusoidal and piecewise affine input function. The estimation error is always contained inside its bound: its absolute value is never greater than the optimal equalized filtering level μ_{opt}. We can observe that typically the behaviour of the filter is better when the input function is sinusoidal, probably because it is smoother than the piecewise affine one, and because the filter has often high gains at high frequencies.



(a)

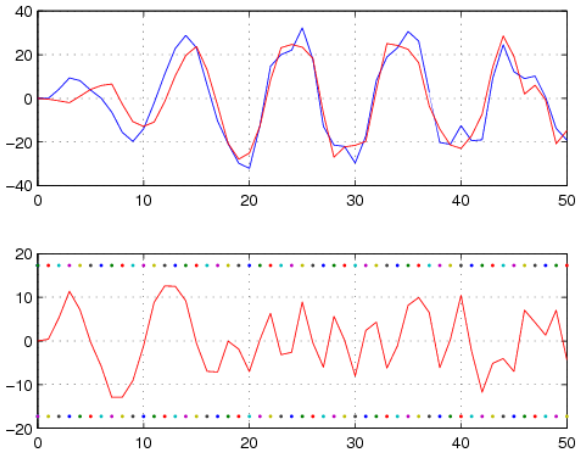


(b)

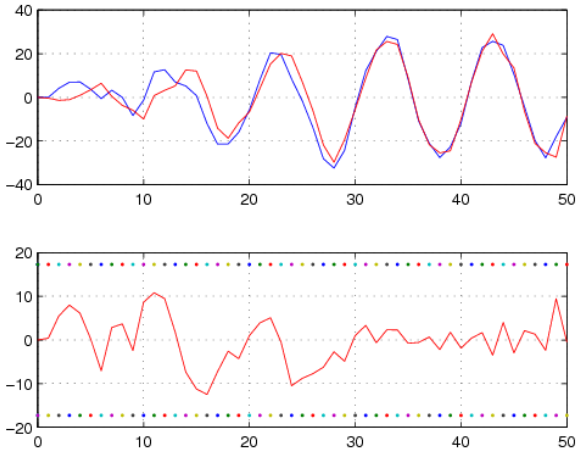


(c)

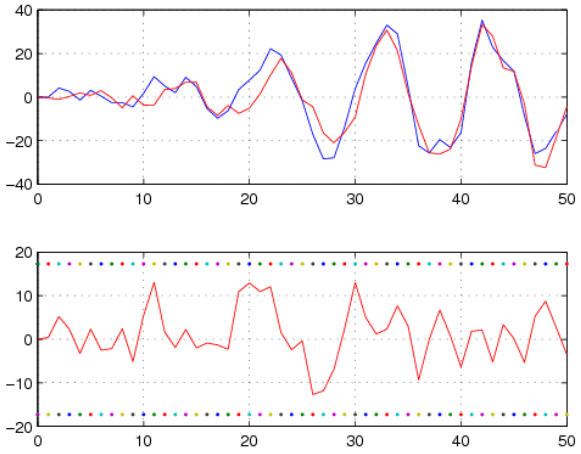
Figure 6.14: Simulations of the filter behaviour for car suspension with sinusoidal input function, in the case $k_1 = 2$, $k_2 = 3$. Above z, in blue, is compared with $\hat{z}$, in red; below, the estimation error is shown and its limits $\pm\mu_{opt}$ are marked



(a)

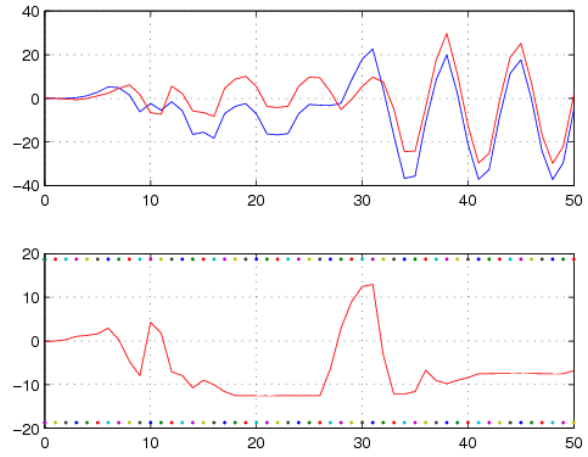


(b)

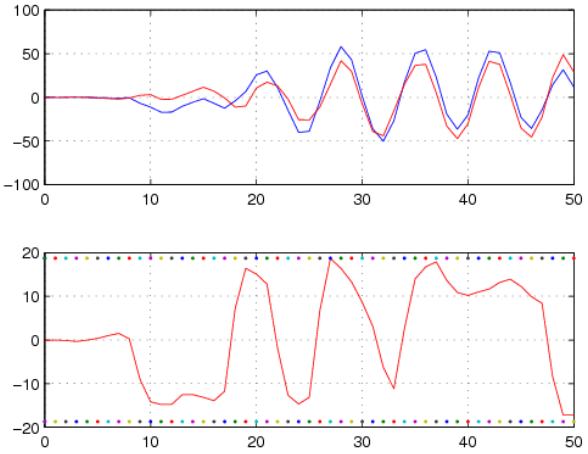


(c)

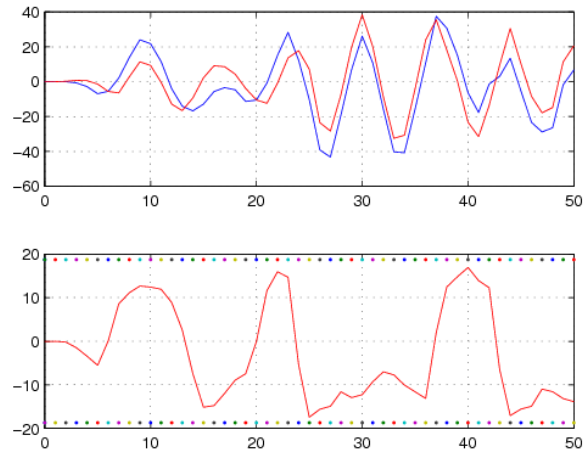
Figure 6.15: Simulations of the filter behaviour for car suspension with sinusoidal input function, in the case $k_1 = 1$, $k_2 = 1$. Above z, in blue, is compared with $\hat{z}$, in red; below, the estimation error is shown and its limits $\pm\mu_{opt}$ are marked



(a)

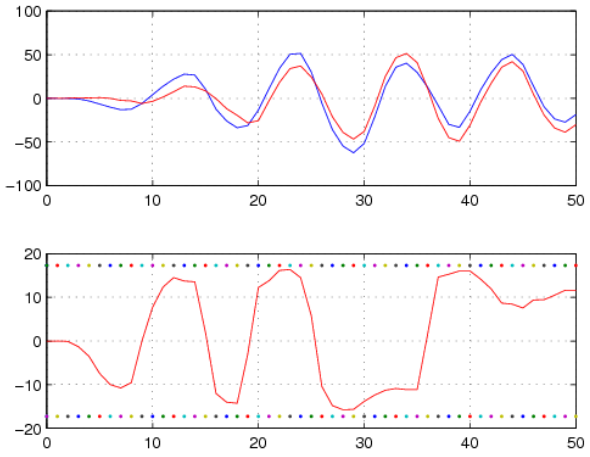


(b)

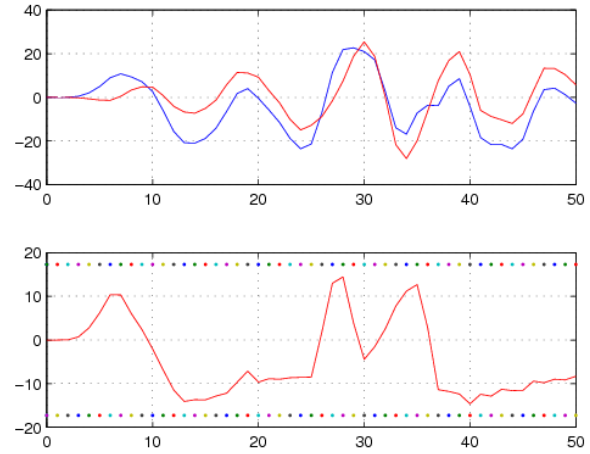


(c)

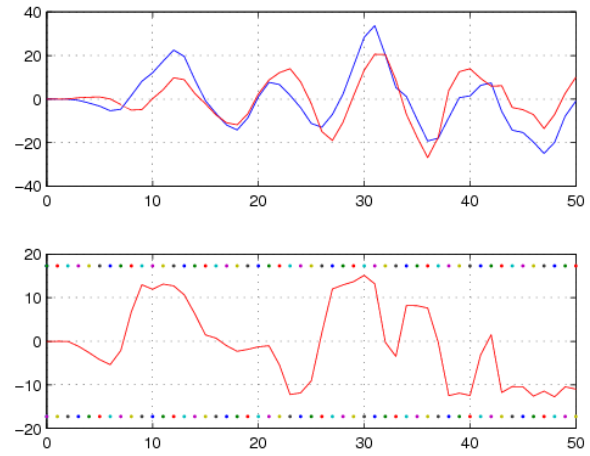
Figure 6.16: Simulations of the filter behaviour for car suspension with piecewise affine input function, in the case $k_1 = 2$, $k_2 = 3$. Above z, in blue, is compared with $\hat{z}$, in red; below, the estimation error is shown and its limits $\pm\mu_{opt}$ are marked



(a)



(b)



(c)

Figure 6.17: Simulations of the filter behaviour for car suspension with piecewise affine input function, in the case $k_1 = 1$, $k_2 = 1$. Above z, in blue, is compared with $\hat{z}$, in red; below, the estimation error is shown and its limits $\pm\mu_{opt}$ are marked

6.3.4 Comparison with Kalman Filter

Consider now the case $k_1 = 2$, $k_2 = 3$; $h = 0.1$, $T_s = 1$, $\beta = 1$ and $\gamma = 10$.
The discrete–time Kalman filter transfer function is

$$\frac{b_{kalman}(\lambda)}{a_{kalman}(\lambda)} \approx \frac{-0.2747\lambda + 0.2291\lambda^2 + 0.6927\lambda^3 + 0.2371\lambda^4}{1 + 0.2504\lambda - 0.2241\lambda^2 + 0.4631\lambda^3 + 0.4112\lambda^4}$$

and its poles are $0.5589 \pm 0.6788j$, $-0.6840 \pm 0.2530j$.

We compare this Kalman filter with the r–equalized filter with $r = 4$, which attains $\mu_{opt} \approx 23.1$ and is

$$\frac{b(\lambda)}{a(\lambda)} \approx \frac{-0.8314 - 1.1603\lambda + 0.6679\lambda^2 + 0.9368\lambda^3 + 0.0000\lambda^4}{1} \quad (7)$$

with

$$c \approx 0 + 1.0275\lambda + 0.4452\lambda^2 - 0.4722\lambda^3 - 0.0000\lambda^4.$$

Its poles are $0.0195 \pm 0.0195j$, $-0.0195 \pm 0.0195j$. They are approximately zero, namely the filter is an FIR.

The Bode diagram of the Kalman filter is compared with that of the r–equalized filter in Fig. 6.18.

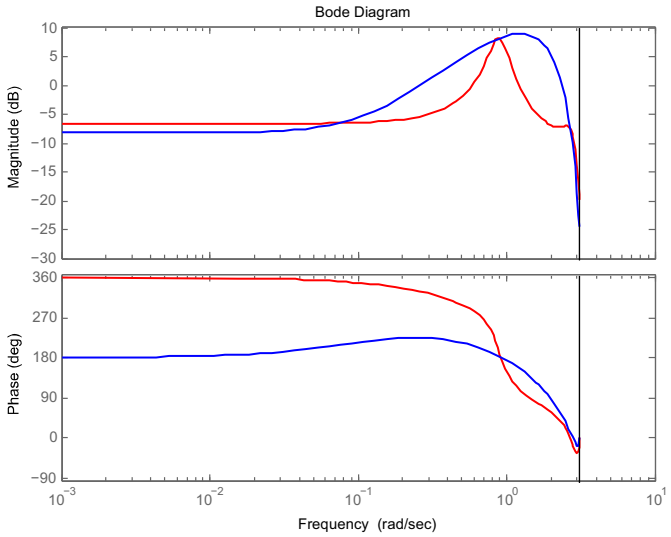


Figure 6.18: Bode diagram of the Kalman filter (in red) and of the equalized filter (in blue) for car suspension

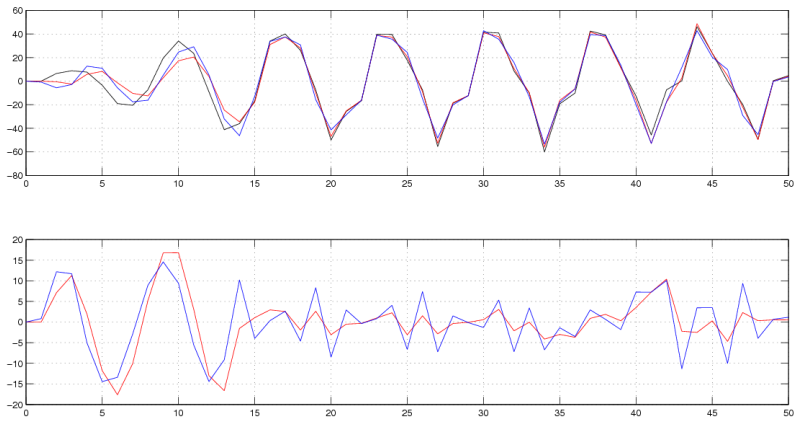
The result of some comparative simulations obtained with sinusoidal input functions and parameters

```
omega_max = 5;
p_change_v = 0.2;
p_change_w = 0.2;
```

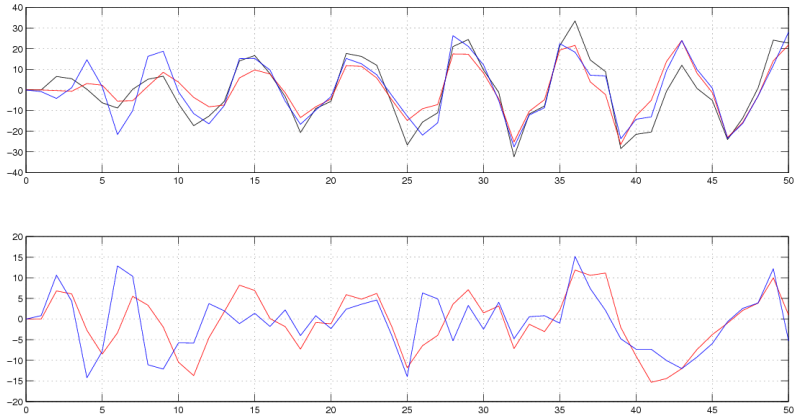
is shown in Fig. 6.19.

From the observation of the frequency response of the two filters, we see that the equalized filter has a slightly sharper cut–off effect. Both filters have a good behaviour, but the worst case estimation error of the equalized filter is slightly smaller in all the experiments we have performed with randomly generated noise.

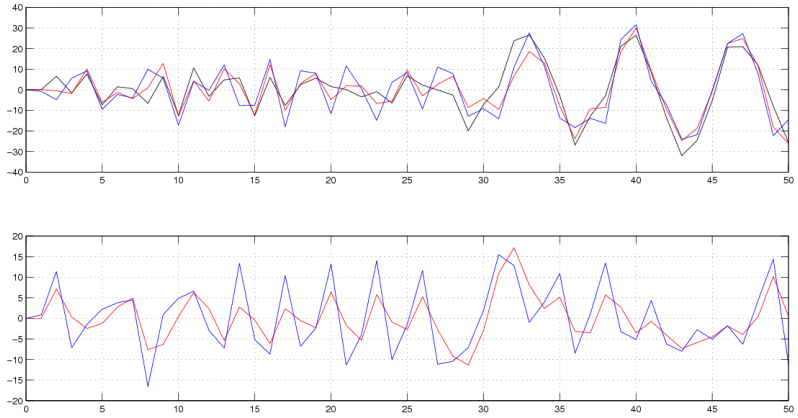
[7] The values a_i with $i > 0$ are not exactly zero, but they are very close to zero



(a)



(b)



(c)

Figure 6.19: Simulations for car suspension with sinusoidal input function. Above z, in black, is compared with $\hat{z}_{eq}$, in blue, and $\hat{z}_{kalman}$, in red; below, the estimation error e_{eq}, in blue, is compared with e_{kalman}, in red

6.4 Masses and Springs System

6.4.1 Model Construction

We consider three masses connected by two springs, as shown in Fig. 6.20, in absence of friction and in presence of a disturbance force v.

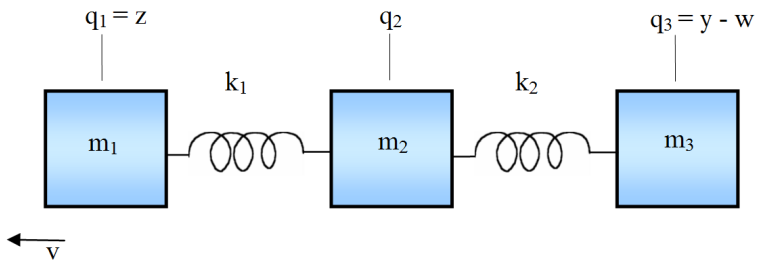


Figure 6.20: Masses and springs system

If we denote by q_i the shift of the mass m_i from its equilibrium position, the equilibrium equations of the forces are

$$\begin{aligned} m_1\ddot{q_1}(t) &= -k_1[q_1(t) - q_2(t)] + v(t) \\ m_2\ddot{q_2}(t) &= -k_1[q_2(t) - q_1(t)] - k_2[q_2(t) - q_3(t)] \\ m_3\ddot{q_3}(t) &= -k_2[q_3(t) - q_2(t)] \end{aligned}$$

Our choice for z and y is

$$\begin{aligned} z(t) &= q_1(t) \\ y(t) &= q_3(t) + w(t) \end{aligned}$$

Under the assumption $m_1 = m_2 = m_3 = 1$, if $x_1 = q_1$, $x_2 = q_2$, $x_3 = q_3$, $x_4 = \dot{q_1}$, $x_5 = \dot{q_2}$, $x_6 = \dot{q_3}$, we obtain the state space realization

$$\begin{aligned} \dot{x} &= Ax + Bv \\ z &= C_z x \\ y &= C_y x + w, \end{aligned}$$

where

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ -k_1 & k_1 & 0 & 0 & 0 & 0 \\ k_1 & -(k_1+k_2) & k_2 & 0 & 0 & 0 \\ 0 & k_2 & -k_2 & 0 & 0 & 0 \end{bmatrix}, \; B = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix},$$

$$C_z = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \; C_y = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}.$$

If we choose $k_1 = k_2 = 1$,

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ -1 & 1 & 0 & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 \end{bmatrix}$$

6.4.2 Filter Synthesis and Simulation

If $T_s = 1$, $\beta = 1$, $\gamma = 1$, $r = 8$, we obtain $\mu_{opt} \approx 5$ and the filter is an FIR

$$\frac{b(\lambda)}{a(\lambda)} \approx \frac{0.4579 + 0.1225\lambda^2 + 0.6719\lambda^3 + 0.3146\lambda^4 - 0.4228\lambda^6 - 0.1684\lambda^7 + 0.0247\lambda^8}{1}(^8),$$

with

$$c \approx 0.4604\lambda + 0.9927\lambda^2 + 0.8281\lambda^3 + 0.3683\lambda^4 + 0.0741\lambda^5 - 0.0014\lambda^7.$$

The Bode diagram of the filter is shown in Fig. 6.21, while Fig. 6.22 shows the result of

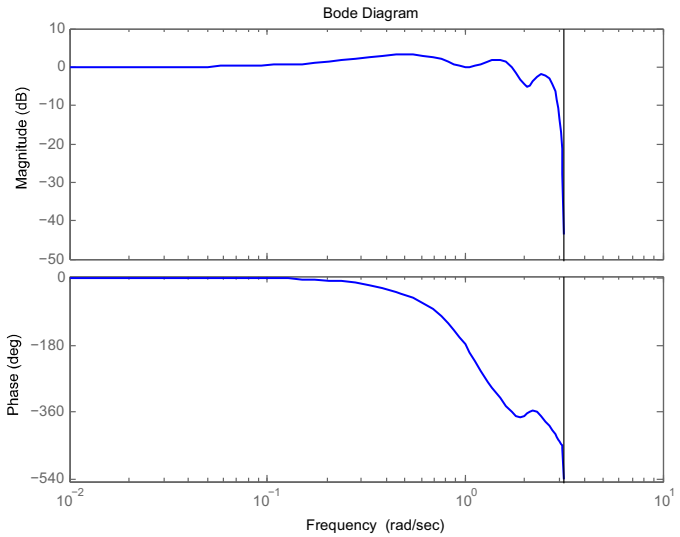


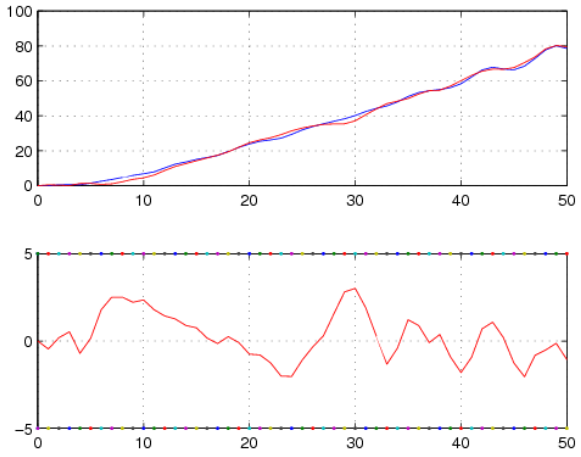
Figure 6.21: Bode diagram of the filter for masses and springs system

some simulations obtained with sinusoidal input functions and parameters

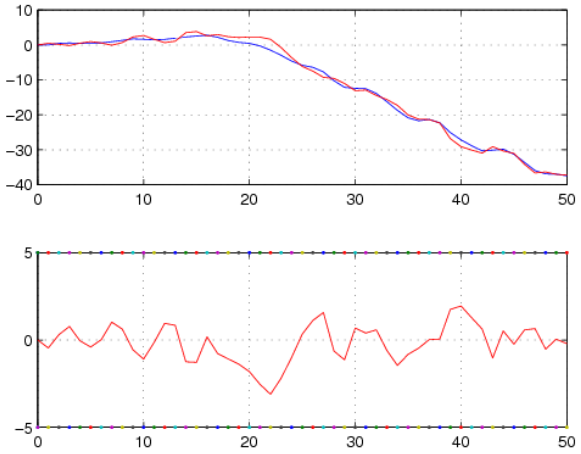
```
omega_max = 5;
p_change_v = 0.2;
p_change_w = 0.2;
```

The behaviour of the filter is satisfactory: $\hat{z}$ estimates accurately z and, as it is guaranteed, the estimation error is never greater in norm than the equalized performance level μ_{opt}.

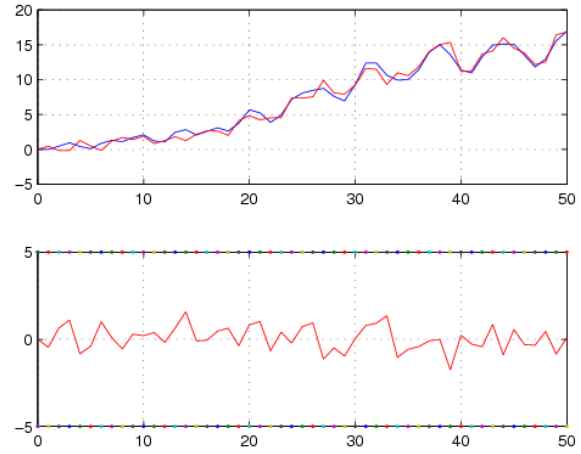
[8] The values a_i with $i > 0$ are not exactly zero, but they are very close to zero



(a)



(b)



(c)

Figure 6.22: Simulations of the filter behaviour for masses and springs system with sinusoidal input function. Above z, in blue, is compared with $\hat{z}$, in red; below, the estimation error is shown and its limits $\pm\mu_{opt}$ are marked

6.4.3 Comparison with Kalman Filter

The discrete–time Kalman filter transfer function is

$$\frac{b_{kalman}(\lambda)}{a_{kalman}(\lambda)} \approx \frac{0.5490\lambda - 0.7669\lambda^2 + 1.4009\lambda^3 - 1.2118\lambda^4 + 0.9062\lambda^5 - 0.5144\lambda^6}{1 - 1.7234\lambda + 2.3620\lambda^2 - 2.3845\lambda^3 + 1.8364\lambda^4 - 0.9879\lambda^5 + 0.2603\lambda^6}$$

and its poles are $-0.1594 \pm 0.9464j$, $0.4055 \pm 0.7143j$, $0.6156 \pm 0.1996j$.

We compare this Kalman filter with the r–equalized filter with $r = 6$, which attains $\mu_{opt} \approx 5.2$ and is

$$\frac{b(\lambda)}{a(\lambda)} \approx \frac{0.3854 + 0.0000\lambda + 0.0231\lambda^2 + 0.7522\lambda^3 + 0.0746\lambda^4 + 0.0000\lambda^5 - 0.4737\lambda^6}{1 - 0.2430\lambda} \quad (9)$$

with

$$c \approx 0 + 0.4605\lambda + 0.8888\lambda^2 + 0.6180\lambda^3 + 0.2212\lambda^4 + 0.0311\lambda^5 + 0.0006\lambda^6.$$

Its poles are 0.2430, $0.0345 \pm 0.0264j$, $-0.0141 \pm 0.0391j$, -0.0408.

The Bode diagram of the Kalman filter is compared with that of the r–equalized filter in Fig. 6.23.

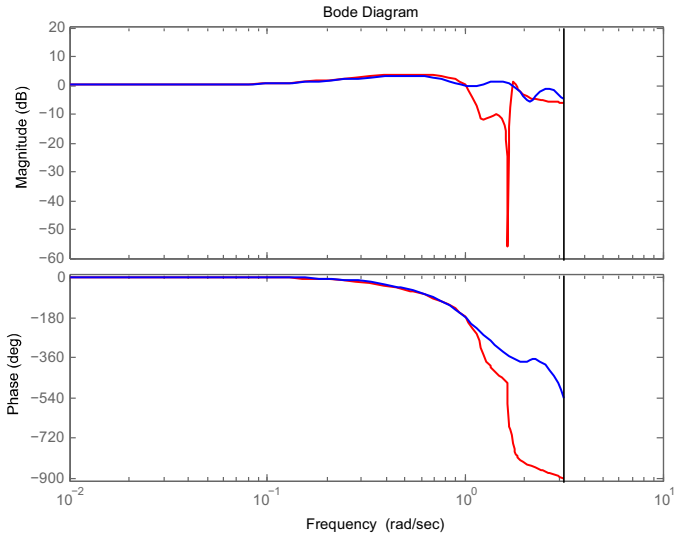


Figure 6.23: Bode diagram of the Kalman filter (in red) and of the equalized filter (in blue) for masses and springs system

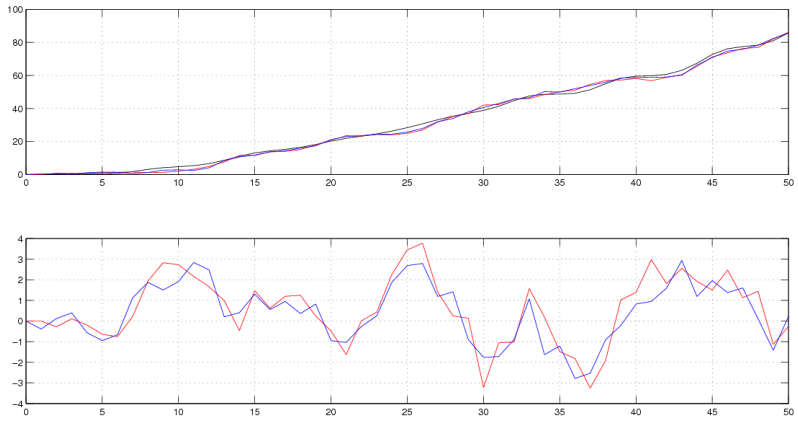
The result of some comparative simulations obtained with sinusoidal input functions and parameters

```
omega_max = 5;
p_change_v = 0.2;
p_change_w = 0.2;
```

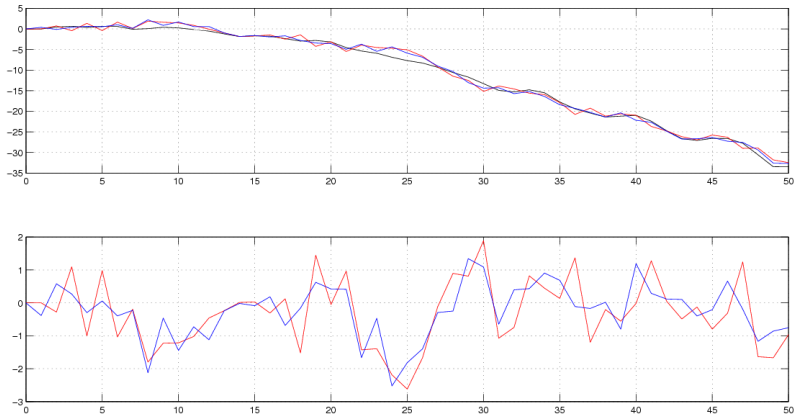
is shown in Fig. 6.24.

The frequency response of the equalized filter is smoother. Both the filters behave well, but, after several experiments with randomly generated noise, we noticed that the equalized filter generally has a smaller worst case error.

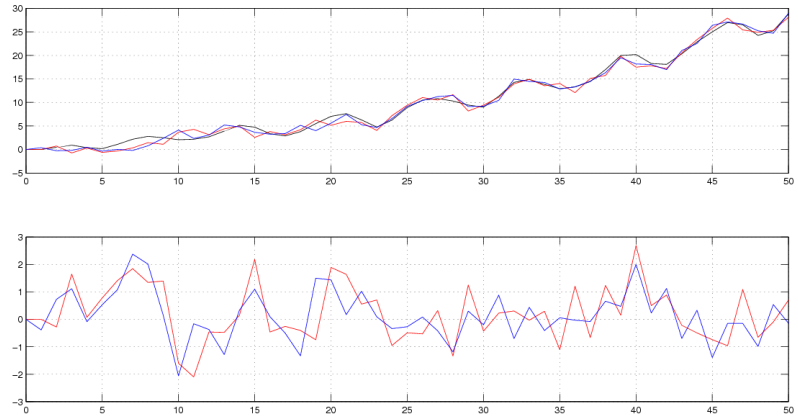
[9] 0.0000 values and a_i with $i > 1$ are not exactly zero, but they are very close to zero



(a)



(b)



(c)

Figure 6.24: Simulations for masses and springs system with sinusoidal input function. Above z, in black, is compared with $\hat{z}_{eq}$, in blue, and $\hat{z}_{kalman}$, in red; below, the estimation error e_{eq}, in blue, is compared with e_{kalman}, in red

Chapter 7

Linear Systems with Known Inputs

In this chapter we show some possible applications of the equalized filter to systems with disturbances in presence of a known input coming from the same channel of the process noise.

We immediately point out a phenomenon that will occur in the following simulations. The norm of the estimation error will not always be contained in a hyperrectangle of size μ_{opt}, not even asymptotically; on the contrary, the worst–case estimation error is likely to be larger than μ_{opt}. As it has been explained in Chapter 3, this effect is due to numerical rounding only: in theory the influence of u should be perfectly cancelled in the estimation error transfer function, but it is not in practice, because the simulation routine works with finite precision. Nevertheless, the filter works well and provides a good estimation $\hat{z}$ of z.

7.1 Simplified Oven

7.1.1 Model Construction

We consider a simplified thermodynamic linear model for an oven. To this aim, we exploit the equivalence between electric circuits and thermal systems.

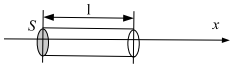


Figure 7.1: Ohm's law

Heat flow: $\dot{Q} = \sigma_T S \frac{\partial T}{\partial x}$	Charge flow: $\dot{q} = -\sigma_e S \frac{\partial V}{\partial x}$, since $I = JS = \frac{1}{\rho_e} S E_x$ and $E_x = -\frac{\partial V}{\partial x}$
Heat in an adiabatic container:	Charge in a capacitor:
$dQ = CdT$	$dq = CdV$
Potential (temperature): $T(x)$	Potential (electrical): $V(x)$

So, if we put

W:	heat flow [W]
C_3:	heating agent thermal capacitance [J/K]
C_4:	oven thermal capacitance [J/K]
R_1:	thermal resistance between heating agent and oven [W/K]
R_2:	thermal resistance between oven and environment [W/K]
T_r:	heating agent temperature [K]
T_f:	oven temperature [K]
T_a:	environment temperature [K],

(a) $T_1 - T_2 = IR$, where $I = \dot{Q}$ and R is thermal resistance

(b) $T_1 - T_2 = \frac{Q}{C}$, where C is thermal capacitance

Figure 7.2: Equivalence between electric circuits and thermal systems

we obtain the equivalent circuit drawn in Fig. 7.3.

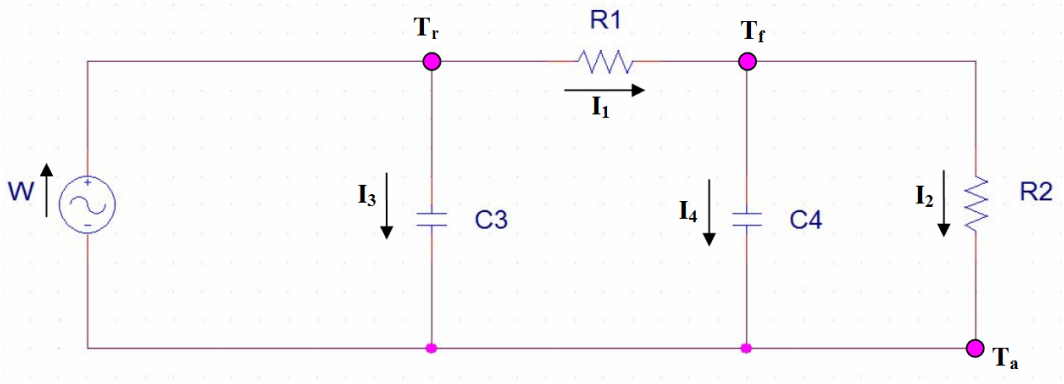


Figure 7.3: Circuit equivalent to the oven thermal system

Time Domain

The equations obtained by applying Kirchhoff's laws are

$$\begin{aligned}
&T_r - T_f = I_1 R_1 \\
&T_f - T_a = I_2 R_2 \\
&W = I_1 + I_3 \\
&I_1 = I_2 + I_4 \\
&R_1 I_1 + \frac{Q_4}{C_4} - \frac{Q_3}{C_3} = 0 \\
&\frac{Q_4}{C_4} = R_2 I_2
\end{aligned}$$

where $I_n = \dot{Q}_n$ and the unknown functions are $T_f(t)$, $T_r(t)$, $Q_3(t)$, $Q_4(t)$, $I_1(t)$, $I_2(t)$. We assume that, after a transient period, equilibrium is reached and that boundary conditions and initial conditions are known: $T_f(0)$, $T_r(0)$, $Q_3(0)$, $Q_4(0)$, $Q_3(\infty)$, $Q_4(\infty)$. With appropriate substitutions in the system we obtain

$$\begin{aligned}
&T_f = T_a + I_2 R_2 \\
&T_r = T_a + I_2 R_2 + I_1 R_1 \\
&I_1 = W - I_3 \\
&I_2 = W - I_3 - I_4 \\
&R_1 (W - I_3) + \frac{Q_4}{C_4} - \frac{Q_3}{C_3} = 0 \\
&\frac{Q_4}{C_4} = R_2 (W - I_3 - I_4) \quad (*)
\end{aligned}$$

and further

$$Q_4 = Q_3 \frac{C_4}{C_3} + R_1 C_4 (I_3 - W) \quad (\diamond)$$

After differentiation we get

$$I_4 = I_3\frac{C_4}{C_3} + R_1C_4\frac{dI_3}{dt}$$

The previous two equations, substituted in (*), give

$$R_2R_1C_4C_3\frac{dI_3}{dt} + [(R_1+R_2)C_3 + R_2C_4]I_3 + Q_3 = (R_1+R_2)C_3W$$

If we define the time constants $\tau_1 = R_1C_3$, $\tau_2 = R_2C_4$, $\tau_3 = R_2C_3$, the equation for Q_3 becomes

$$\tau_1\tau_2\ddot{Q_3} + (\tau_1+\tau_2+\tau_3)\dot{Q_3} + Q_3 = (\tau_1+\tau_3)W$$

The characteristic equation associated to the homogeneous differential equation is

$$\tau_1\tau_2\zeta^2 + (\tau_1+\tau_2+\tau_3)\zeta + 1 = 0$$

whose solutions are

$$\zeta_{1,2} = \frac{-(\tau_1+\tau_2+\tau_3) \pm \sqrt{(\tau_1+\tau_2+\tau_3)^2 - \tau_1\tau_2}}{\tau_1\tau_2}$$

and a particular solution of the non homogeneous differential equation is

$$Q_{30} = (\tau_1+\tau_3)W,$$

so that the general solution is

$$\begin{aligned} Q_3(t) &= Q_{31}e^{\zeta_1 t} + Q_{32}e^{\zeta_2 t} + (\tau_1+\tau_3)W \\ I_3(t) &= Q_{31}\zeta_1 e^{\zeta_1 t} + Q_{32}\zeta_2 e^{\zeta_2 t} \end{aligned}$$

If we substitute these equations into ($\diamond$) and we define a fourth time constant $\tau_4 = R_1C_4$, we obtain

$$\begin{aligned} Q_4(t) &= Q_{31}(\tau_4\zeta_1 + \frac{C_4}{C_3})e^{\zeta_1 t} + Q_{32}(\tau_4\zeta_2 + \frac{C_4}{C_3})e^{\zeta_2 t} + \tau_2 W \\ I_4(t) &= Q_{31}\zeta_1(\tau_4\zeta_1 + \frac{C_4}{C_3})e^{\zeta_1 t} + Q_{32}\zeta_2(\tau_4\zeta_2 + \frac{C_4}{C_3})e^{\zeta_2 t} \end{aligned}$$

Finally we substitute the currents in the equation for T_f and we obtain

$$T_f(t) = T_a + WR_2 - R_2[Q_{31}\zeta_1(1+\tau_4\zeta_1 + \frac{C_4}{C_3})e^{\zeta_1 t} + Q_{32}\zeta_2(1+\tau_4\zeta_2 + \frac{C_4}{C_3})e^{\zeta_2 t}]$$

If the oven is good, it has the following characteristics

- $C_4 > C_3$: a large thermal inertia on the heating agent guarantees stability, because the system to control (the oven itself) follows the controlled object (the heater);
- $R_2 >> R_1$: few losses towards the environment.

In these conditions the oven controlled with

$$W = \frac{T_f^* - T_a}{R_2}$$

reaches the required temperature T_f^*, even though the transient can be quite slow.

Laplace Transform

The study of the system in the time domain explains its behaviour. In order to synthesize the filter we need to work with Laplace transform.

We choose as z, output to estimate, the temperature of the oven and as y, measured output, the temperature of the heating agent. This is motivated by the fact that it could be easier to put a temperature sensor on the heating agent or to obtain its temperature from a resistance measure (since resistivity varies with temperature).

From the analysis of the circuit, we express by means of the Laplace transform the transfer functions from W to $T_r - T_a$ (which is $\frac{N}{d}$) and from W to $T_f - T_a$ (which is $\frac{M}{d}$).

$$\begin{aligned}(R_1 + C_4 \parallel R_2) \parallel C_3 &= (R_1 + \frac{R_2}{1 + sC_4R_2}) \parallel C_3 \\ &= (\frac{1 + sC_4R_2}{R_1 + R_2 + sC_4R_2R_1} + sC_3)^(- 1) \\ &= \frac{R_1 + R_2 + sC_4R_2R_1}{1 + sC_4R_2 + sC_3(R_1 + R_2) + s^2C_4C_3R_2R_1}\end{aligned}$$

So the transfer function from W to $T_r - T_a$ is

$$T_r - T_a = \frac{R_1 + R_2 + sC_4R_2R_1}{1 + sC_4R_2 + sC_3(R_1 + R_2) + s^2C_4C_3R_2R_1}W$$

and

$$\begin{aligned}T_f - T_a &= (T_r - T_a)\frac{\frac{R_2}{1+sC_4R_2}}{R_1 + \frac{R_2}{1+sC_4R_2}} = (T_r - T_a)\frac{R_2}{R_1 + R_2 + sC_4R_2R_1} \\ &= \frac{R_2}{1 + sC_4R_2 + sC_3(R_1 + R_2) + s^2C_4C_3R_2R_1}W\end{aligned}$$

is the transfer function from W to $T_f - T_a$.

7.1.2 Filter Synthesis and Simulation

We choose the parameters of the system

$$\begin{aligned}R_1 &= 1\,\mathrm{W/K} \\ R_2 &= 1000\,\mathrm{W/K} \\ C_3 &= 0.001\,\mathrm{J/K} \\ C_4 &= 0.01\,\mathrm{J/K} \\ T_f^* &= 524\,\mathrm{K} \\ T_a &= 294\,\mathrm{K}\end{aligned}$$

If $T_s = 1$, $\beta = 1$, $\gamma = W/20$, $r = 3$, we obtain $\mu_{opt} \approx 1$; the two components of the filter are

$$\frac{b(\lambda)}{a(\lambda)} \approx \frac{0.9897 + 0.0056\lambda + 0.0001\lambda^2 + 0.0000\lambda^3}{1 - 0.0038\lambda - 0.0001\lambda^2 - 0.0000\lambda^3}$$

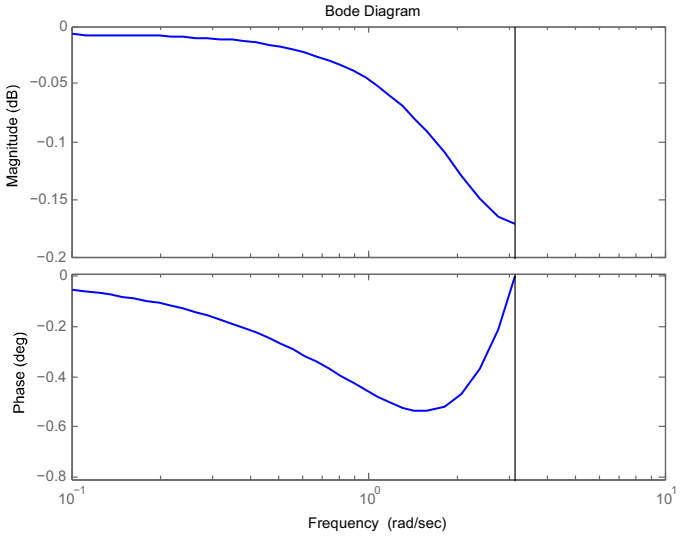
and

$$\frac{c(\lambda)}{a(\lambda)} \approx \frac{0 - 0.0132\lambda - 0.0097\lambda^2 + 0.0024\lambda^3}{1 - 0.0038\lambda - 0.0001\lambda^2 - 0.0000\lambda^3}.$$

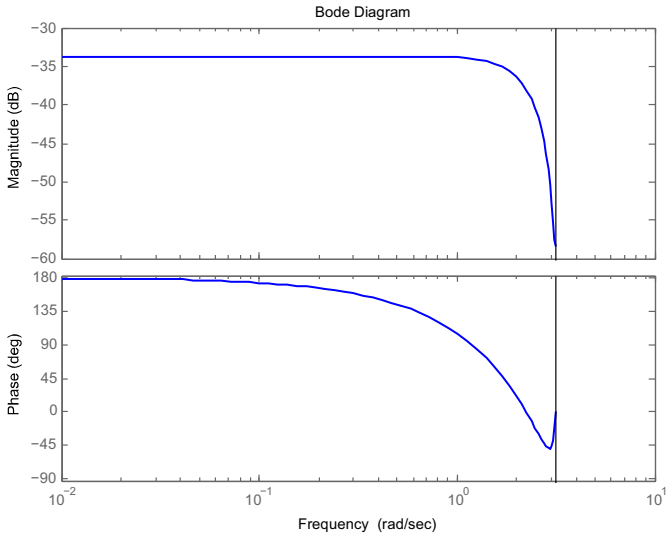
The poles are 0.0304, $-0.0133 \pm 0.0240j$.

The Bode diagram of the two components of the filter is shown in Fig. 7.4 and Fig. 7.5 shows the result of some simulations obtained with sinusoidal input functions and parameters

```
omega_max = 4;
p_change_v = 0.8;
p_change_w = 0.5;
```

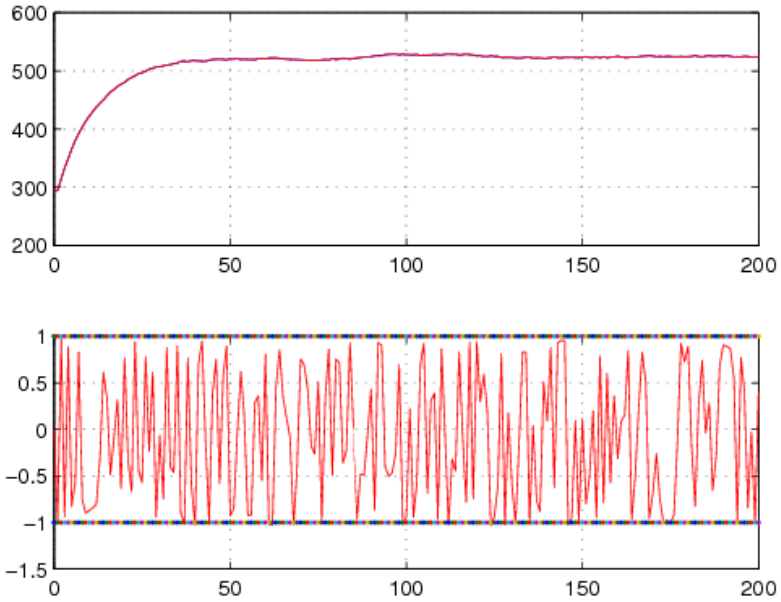


(a) $\frac{b(\lambda)}{a(\lambda)}$

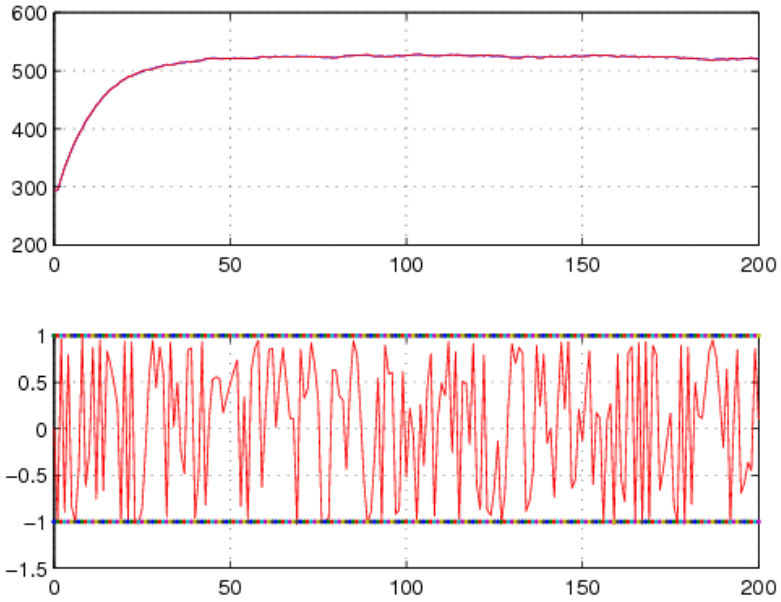


(b) $\frac{c(\lambda)}{a(\lambda)}$

Figure 7.4: Bode diagrams of the filter for the oven system



(a)



(b)

Figure 7.5: Simulations of the filter behaviour for the oven system with sinusoidal input function. Above z, in blue, is compared with $\hat{z}$, in red; below, the estimation error is shown and $\pm\mu_{opt}$ are marked

We can observe that the oven reaches rather quickly the desired temperature of 524 K and then maintains it.

7.1.3 Remarks

The filter works very well: the estimation error is small, it is even included in the close interval $\pm\mu_{opt}$. This happens because thermal systems are usually very manageable. The system we considered is a *minimal phase system*, i.e. in its transfer functions not only the denominator roots, but also the numerator roots are *stable*. This property makes filtering easier. Consider the block scheme in Fig. 7.6.

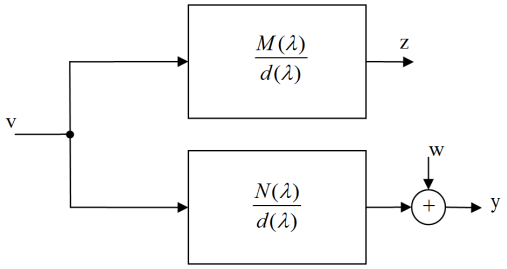


Figure 7.6: Block scheme - minimal phase system

Since it is possible to invert the relation

$$y = w + \frac{N}{d}v \quad \Longleftrightarrow \quad v = (y - w)\frac{d}{N},$$

we can write

$$z = \frac{M}{d}v = \frac{M}{d}\frac{d}{N}(y - w) = \frac{M}{N}y - \frac{M}{N}w,$$

so that

$$\hat{z} = \frac{M}{N}y$$

is a good estimation for z if the disturbance w is not too large. Such a filter is stable: the roots of the polynomial N are stable, because the system has minimal phase.

7.1.4 Comparison with Kalman Filter

The discrete–time Kalman filter transfer function is

$$\frac{b_{kalman}(\lambda)}{a_{kalman}(\lambda)} \approx \frac{+0.9043\lambda + 0.0000\lambda^2}{1 - 0.0087\lambda + 0.0000\lambda^2}$$

and its poles are 0.0087 and 0.0000.

We compare this Kalman filter with the r–equalized, $r = 2$, which attains $\mu_{opt} \approx 1.05$:

$$\frac{b(\lambda)}{a(\lambda)} \approx \frac{0.9896 + 0.0093\lambda + 0.0000\lambda^2}{1 - 0.0000\lambda - 0.0001\lambda^2}$$

and

$$\frac{c(\lambda)}{a(\lambda)} \approx \frac{0 - 0.0023\lambda - 0.0000\lambda^2}{1 - 0.0000\lambda - 0.0001\lambda^2}(^1).$$

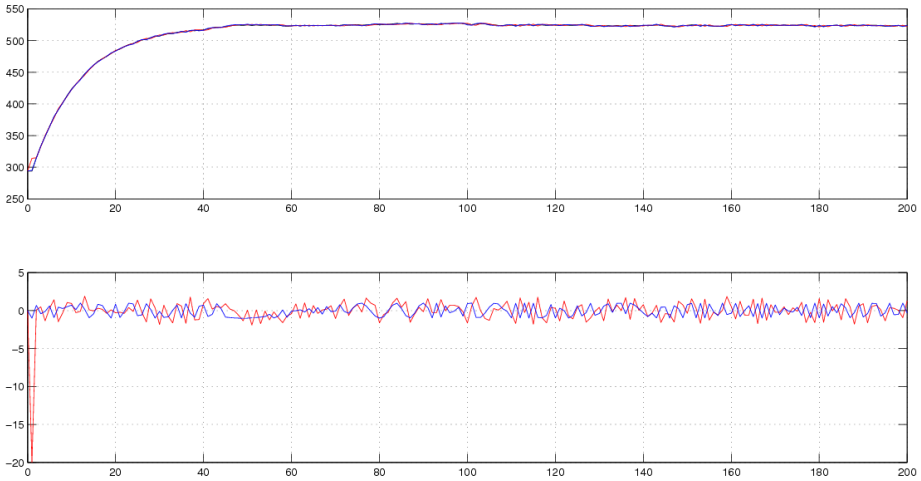
The poles are ± 0.0086.

The result of a comparative simulation obtained with sinusoidal input functions and parameters

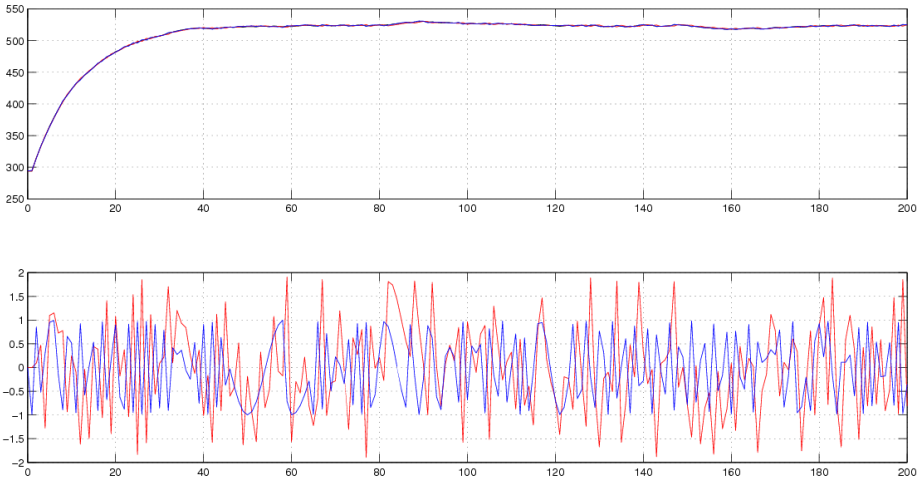
[1] All 0.0000 values in a, b and c are not exactly zero, but they are very close to zero

```
omega_max = 5;
p_change_v = 0.2;
p_change_w = 0.2;
```

is shown in Fig. 7.7 and the Bode diagrams are shown in Fig. 7.8.



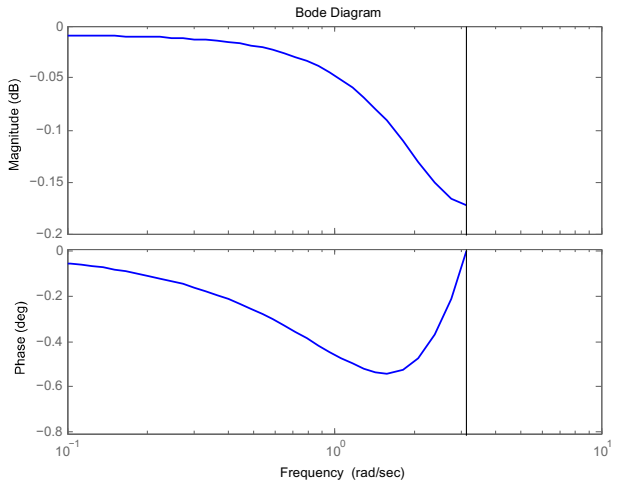
(a) *First simulation*



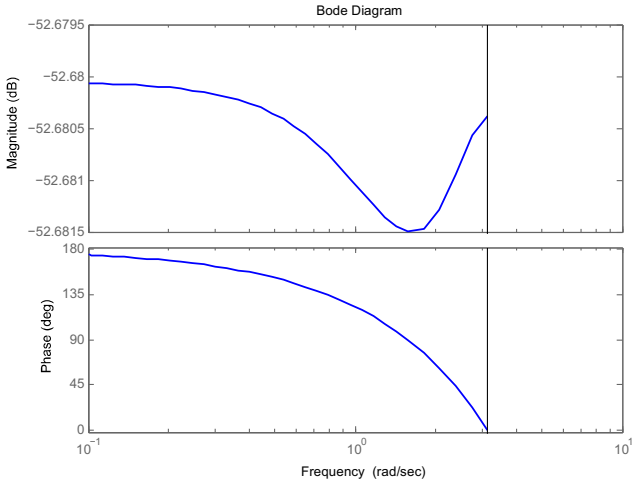
(b) *Second simulation*

Figure 7.7: Simulations for the oven system with sinusoidal input function. Above z, in black, is compared with $\hat{z}_{eq}$, in blue, and $\hat{z}_{kalman}$, in red; below, the estimation error e_{eq}, in blue, is compared with e_{kalman}, in red

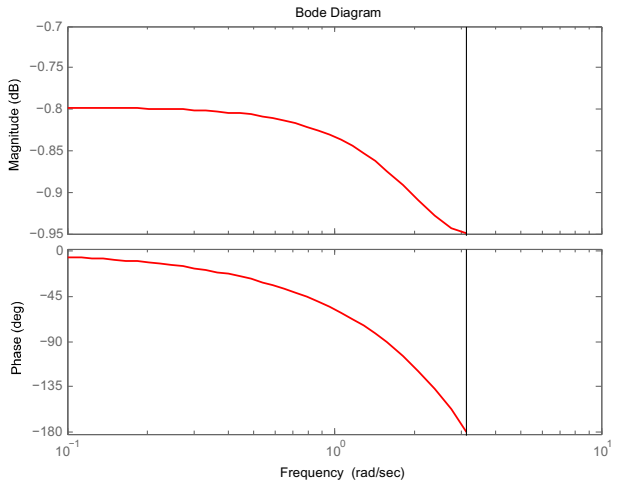
The first sample of the Kalman filter estimation error is much greater than all the others, which are similar to (but a little greater than) the estimation error of the equalized filter. This may happen because the Kalman filter needs some steps before stabilising and reaching steady–state. At the first step, the presence of control in the Kalman filter system leads to overestimate the output. However, if we add control only in the following steps, then the Kalman filter estimate improves: all the samples have the same order of magnitude and both the filters behave very well. Anyway the worst case estimation error of the equalized filter is always better.



(a) $\frac{b(\lambda)}{a(\lambda)}$



(b) $\frac{c(\lambda)}{a(\lambda)}$



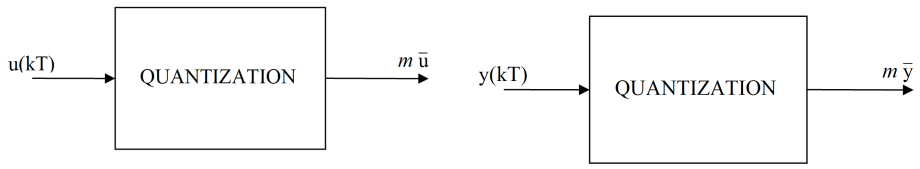
(c) $\frac{b_{kal}(\lambda)}{a_{kal}(\lambda)}$

Figure 7.8: Bode diagram of the Kalman filter (in red) and of the equalized filter (in blue) for the oven system

7.2 Second Order Quantized System

7.2.1 Model Construction

In a system with quantized input u and quantized output y, the data that can be furnished in input and the data provided as output can belong only to a numerable set of values (as shown in Fig. 7.9) which are multiples of a constant value, called quantization step: if a



(a) *Input* (b) *Output*

Figure 7.9: Quantization

sequence of random values $u(kT)$ is supplied as input, each value must be replaced by the nearest multiple of the quantization step $m\ \overline{u}$, with $m \in \mathbb{Z}$. Fig. 7.10 shows how quantization transforms a given continuous function into a piecewise constant function whose values belong to the set $\{m\ \overline{q} | m \in \mathbb{Z}\}$, where $\overline{q}$ is the quantization step.

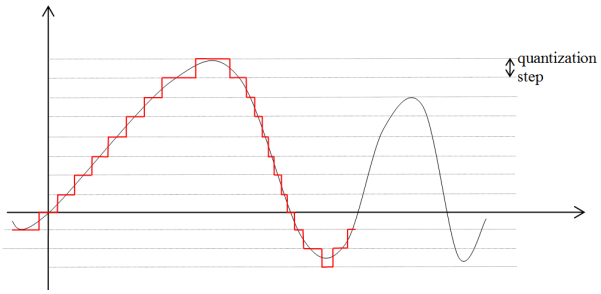


Figure 7.10: A continuous quantity (in black) converted into a quantized one (in red)

Of course this process causes a distortion of the signal: the values supplied as input are not processed by the system, which receives the quantized signal. The difference between the original signal and the quantized one is called quantization noise: as it is easy to see from Fig. 7.10, its maximum width is $\overline{q}/2$, where $\overline{q}$ is the quantization step.

Quantization noise can be modelled as a disturbance v added to the signal. Such a disturbance, due to the previous consideration, is limited in norm as follows:

$$\|v(t)\| \leq \overline{q}/2$$

Consider the system with both quantized input and quantized output in Fig. 7.11: u is the supplied input, v represents the quantization noise; the output generated by the system can potentially assume any value and w represents the quantization noise in output, so that the output furnished outside, y, is quantized.

We aim to analyse now a system with continuous–time transfer function from input u to the output to estimate, z,

$$\frac{s}{s^2 + \omega^2}$$

and continuous–time transfer function from input u to measured output y

$$\frac{1}{s^2 + \omega^2},$$

as it is shown in Fig. 7.12.

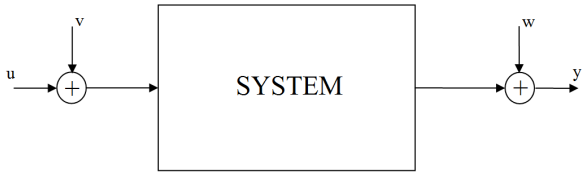


Figure 7.11: System with quantized input and quantized output

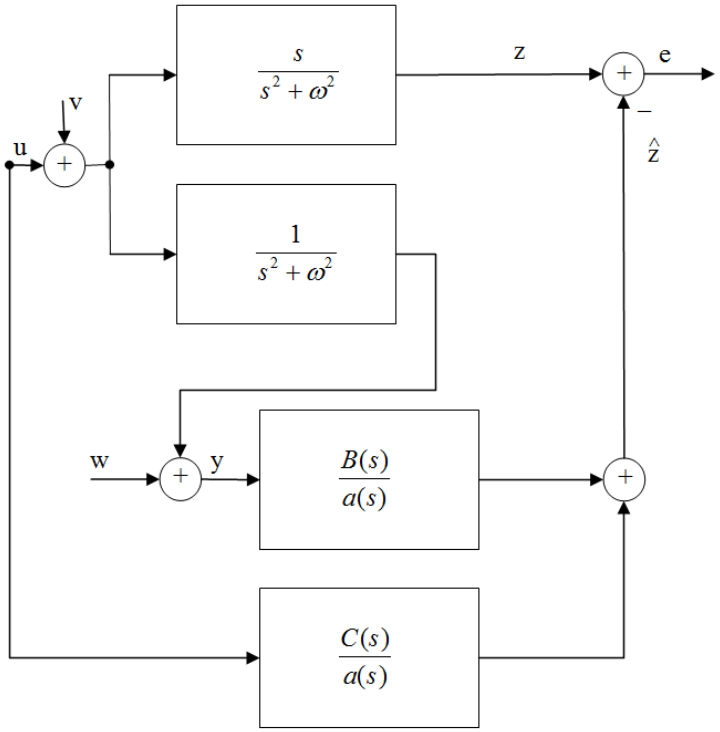


Figure 7.12: Block scheme of the second order quantized system

We assume that both input and output are quantized. Input quantization noise and output quantization noise are modelled, respectively, as a disturbance v added to u and a disturbance w contained in y. Input quantization step is 2γ and output quantization step is 2β, so that the maximum quantization error (i.e. the maximum norm of disturbances v and w) is respectively γ (input) and β (output).

7.2.2 Filter Synthesis and Simulation

If $\omega = 2$, $T_s = 0.2$, $\beta = 1$, $\gamma = 1$, $r = 3$, we obtain $\mu_{opt} \approx 3.1$; the two components of the filter are

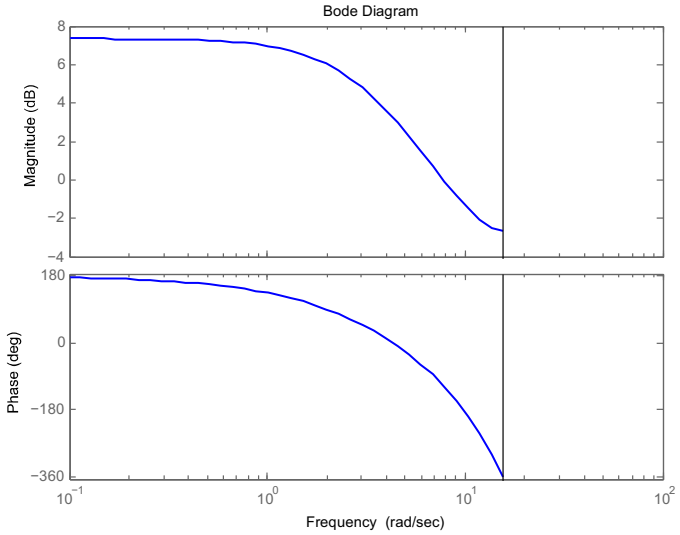
$$\frac{b(\lambda)}{a(\lambda)} \approx \frac{0.0000 - 0.0000\lambda - 0.0000\lambda^2 - 1.1189\lambda^3}{1 - 0.5200\lambda + 0.0000\lambda^2 - 0.0000\lambda^3}$$

and

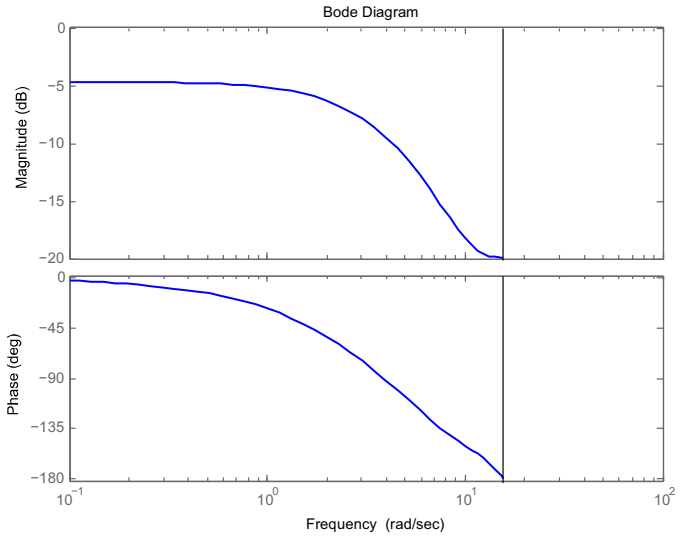
$$\frac{c(\lambda)}{a(\lambda)} \approx \frac{0 + 0.1947\lambda + 0.0627\lambda^2 + 0.0221\lambda^3}{1 - 0.5200\lambda + 0.0000\lambda^2 - 0.0000\lambda^3}([2]).$$

The poles are 0.5200, ± 0.0001. The Bode diagram of the two components of the filter is shown in Fig. 7.13 and Fig. 7.14 shows the result of some simulations obtained with sinusoidal input functions and parameters

[2] All 0.0000 values in a and b are not exactly zero, but they are very close to zero



(a) $\frac{b(\lambda)}{a(\lambda)}$



(b) $\frac{c(\lambda)}{a(\lambda)}$

Figure 7.13: Bode diagrams of the filter for the second order quantized system

```
omega_max = 5;
ampiezza_max = 10; % maximum amplitude value of u
p_change_u = 0.2;
```

It is evident that the estimation error is not always included in the interval $\pm\mu_{opt}$: as already discussed, this is due the non–exact cancellation of the known input effect in the estimation error transfer function. Anyway, the behaviour of the filter is good and $\hat{z}$ follows well z; the estimation error can be considered small in proportion to the amplitude of the signal.

7.2.3 Comparison with Kalman Filter

The discrete–time Kalman filter transfer function is given by

$$\frac{b_{kalman}(\lambda)}{a_{kalman}(\lambda)} \approx \frac{-0.0521\lambda - 0.0228\lambda^2}{1 - 1.7512\lambda + 0.9061\lambda^2}$$

and its poles are $0.8756 \pm 0.3734j$.

We compare this Kalman filter with the r–equalized, $r = 2$, which attains $\mu_{opt} \approx 4.4$:

$$\frac{b(\lambda)}{a(\lambda)} \approx \frac{0.0000 - 0.0000\lambda - 0.8462\lambda^2}{1 - 0.7562\lambda - 0.0000\lambda^2}$$

and

$$\frac{c(\lambda)}{a(\lambda)} \approx \frac{0 + 0.1947\lambda + 0.0167\lambda^2}{1 - 0.7562\lambda - 0.0000\lambda^2}\,(^3).$$

The poles are 0.7562, -0.0000.

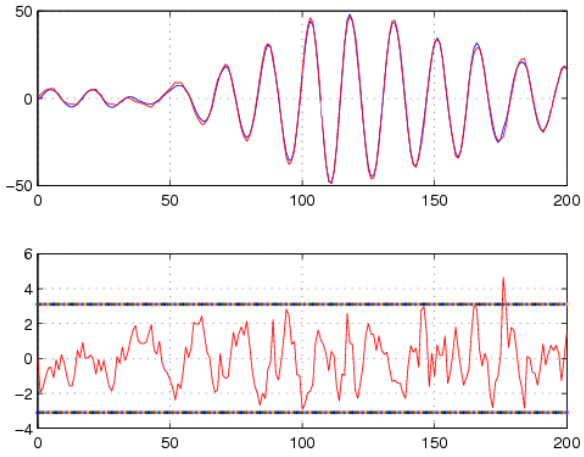
The Bode diagrams are shown in Fig. 7.15 and the result of some comparative simulations obtained with sinusoidal input functions and parameters

```
omega_max = 4;
ampiezza_max = 10; % maximum amplitude value of u
p_change_u = 0.2;
```

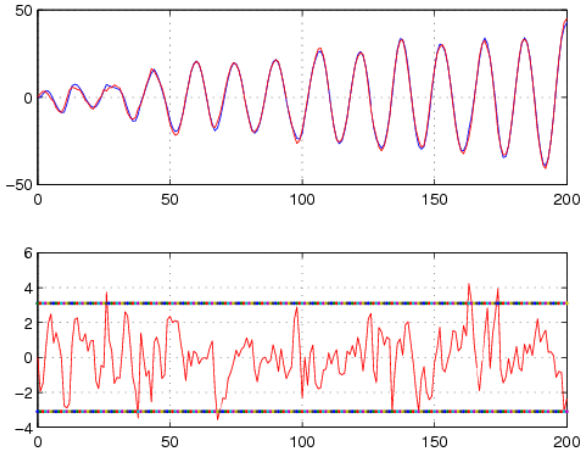
is shown in Fig. 7.16.

Both the filters behave well, but several experiments with randomly generated noise show that the equalized filter usually has a noticeably smaller worst case error.

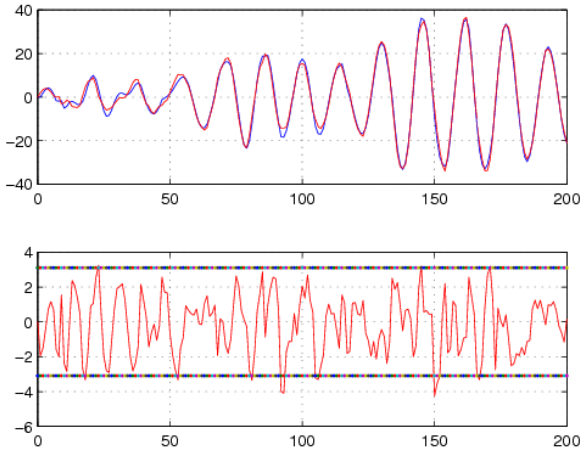
[3] All 0.0000 values in a and b are not exactly zero, but they are very close to zero



(a)

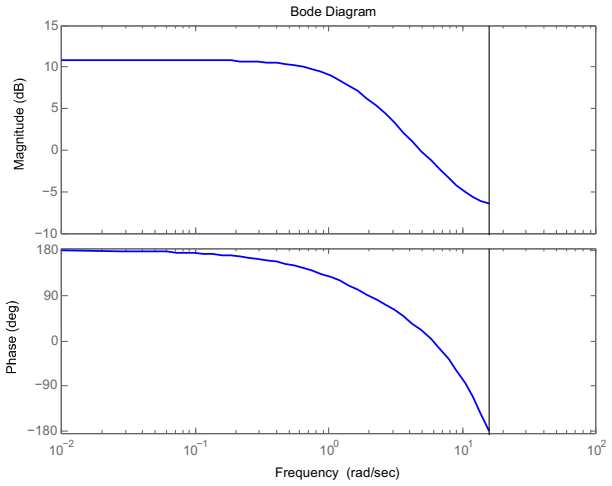


(b)

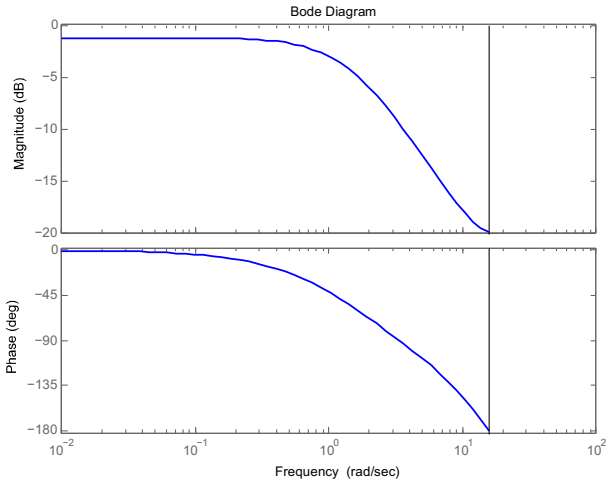


(c)

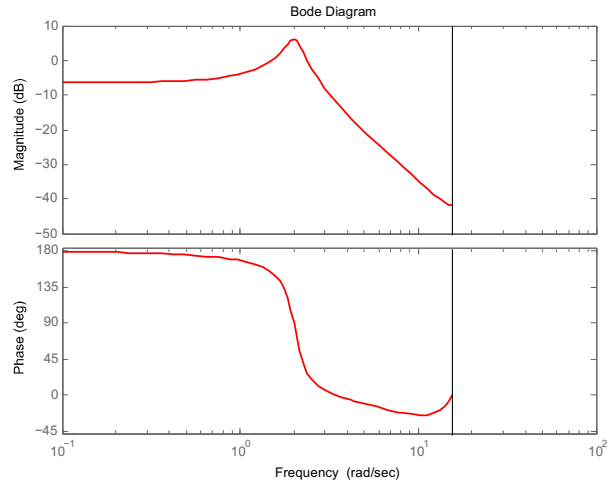
Figure 7.14: Simulations of the filter behaviour for second order quantized system with sinusoidal input function. Above z, in blue, is compared with $\hat{z}$, in red; below, the estimation error is shown and $\pm\mu_{opt}$ are marked



(a) $\frac{b(\lambda)}{a(\lambda)}$

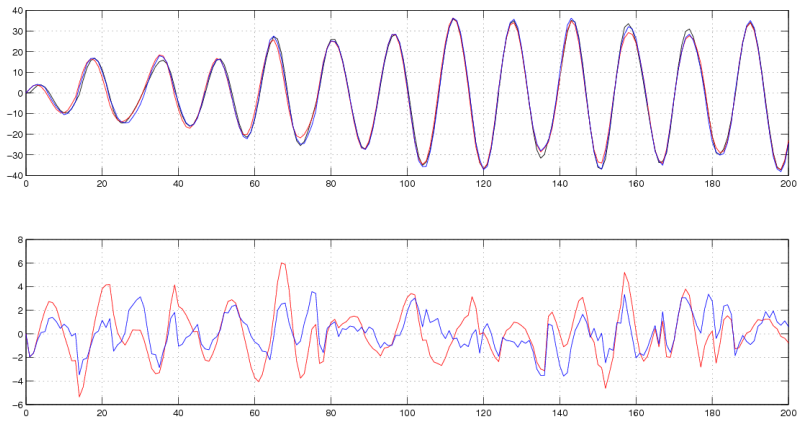


(b) $\frac{c(\lambda)}{a(\lambda)}$

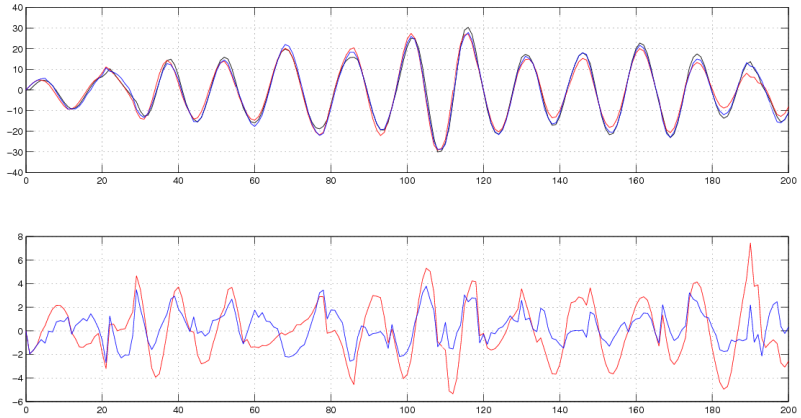


(c) $\frac{b_{kal}(\lambda)}{a_{kal}(\lambda)}$

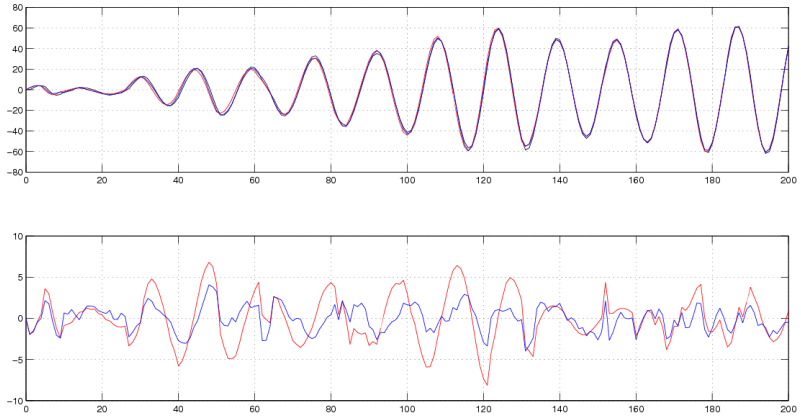
Figure 7.15: Bode diagram of the Kalman filter (in red) and of the equalized filter (in blue) for the second order quantized system



(a)



(b)



(c)

Figure 7.16: Simulations for second order quantized system with sinusoidal input function. Above z, in black, is compared with $\hat{z}_{eq}$, in blue, and $\hat{z}_{kalman}$, in red; below, the estimation error e_{eq}, in blue, is compared with e_{kalman}, in red

7.3 Fourth Order Quantized System

7.3.1 Model Construction

With the same hypotheses of the previous section, we aim now to analyse a system of the fourth order. Its continuous–time transfer function from input u to the output to estimate, z, is

$$\frac{s}{(s^2+\omega_1^2)(s^2+\omega_2^2)}$$

and its continuous–time transfer function from input u to measured output y is

$$\frac{1}{(s^2+\omega_1^2)(s^2+\omega_2^2)},$$

as it is shown in Fig. 7.17.

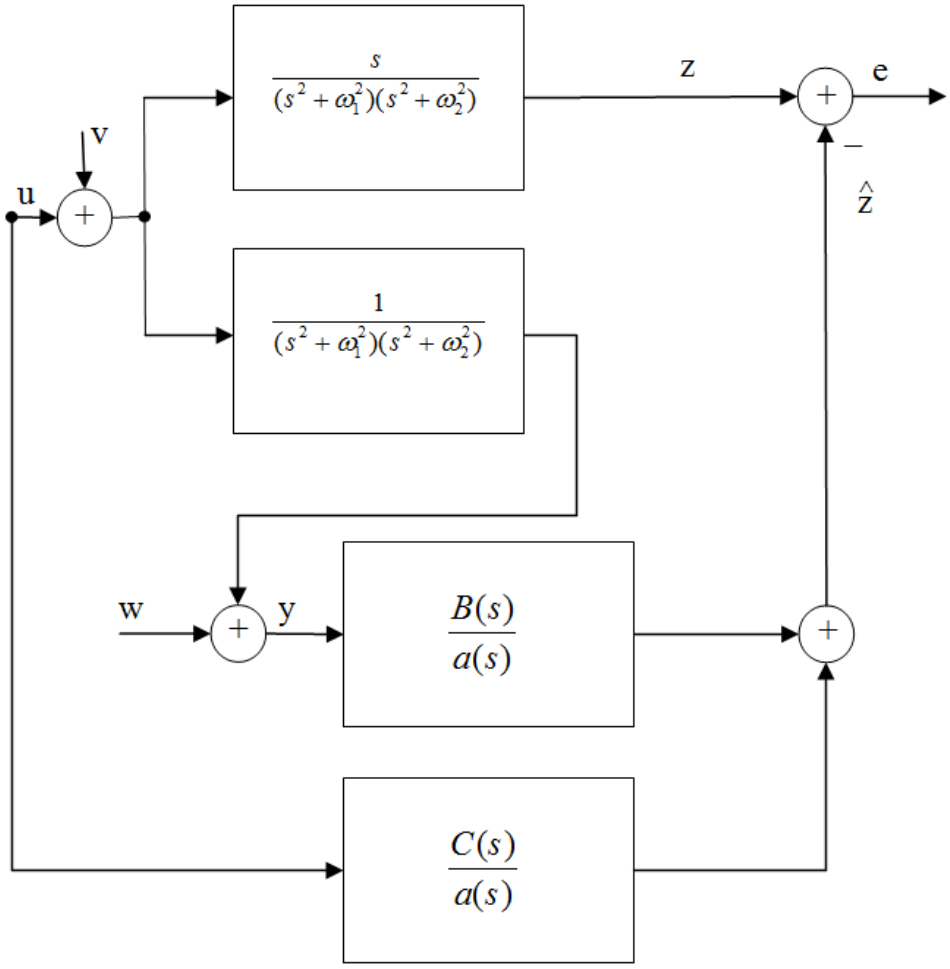


Figure 7.17: Block scheme of the fourth order quantized system

7.3.2 Filter Synthesis and Simulation

If we take $\omega_1 = 2$, $\omega_2 = 4$, $T_s = 0.2$, $\beta = 1$, $\gamma = 1$, $r = 6$, we obtain $\mu_{opt} \approx 12.42$; the two components of the filter are

$$\frac{b(\lambda)}{a(\lambda)} \approx \frac{2.6939 - 0.0023\lambda - 5.2751\lambda^2 + 0.0001\lambda^3 + 2.7010\lambda^4 + 0.0024\lambda^5 - 1.4183\lambda^6}{1 - 0.0034\lambda^6}$$

and

$$\frac{c(\lambda)}{a(\lambda)} \approx \frac{0 + 0.0011\lambda + 0.0054\lambda^2 + 0.0074\lambda^3 + 0.0045\lambda^4 + 0.0013\lambda^5 + 0.0001\lambda^6}{1 - 0.0034\lambda^6}([4]).$$

The poles are 0.3882, -0.3879, $0.1940 \pm 0.3362j$, $-0.1941 \pm 0.3359j$.

The Bode diagram of the two components of the filter is shown in Fig. 7.18 and the result of some simulations obtained with sinusoidal input functions and parameters

```
omega_max = 10;
ampiezza_max = 1000; % maximum amplitude value of u
p_change_u = 0.5;
```

is shown in Fig. 7.19.

7.3.3 Another Filter Synthesis and Simulation

If instead $\omega_1 = 4$, $\omega_2 = 4$, with $T_s = 0.2$, $\beta = 1$, $\gamma = 1$, $r = 6$, we obtain $\mu_{opt} \approx 11$; the two components of the filter are

$$\frac{b(\lambda)}{a(\lambda)} \approx \frac{1.0687 - 0.0414\lambda - 6.2011\lambda^2 + 0.0065\lambda^3 + 0.8584\lambda^4 + 0.0717\lambda^5 - 2.1294\lambda^6}{1 - 0.0011\lambda + 0.0004\lambda^2 + 0.0021\lambda^3 + 0.0004\lambda^4 - 0.0015\lambda^5 - 0.0279\lambda^6}$$

and

$$\frac{c(\lambda)}{a(\lambda)} \approx \frac{0 + 0.0012\lambda + 0.0060\lambda^2 + 0.0085\lambda^3 + 0.0057\lambda^4 + 0.0019\lambda^5 + 0.0002\lambda^6}{1 - 0.0011\lambda + 0.0004\lambda^2 + 0.0021\lambda^3 + 0.0004\lambda^4 - 0.0015\lambda^5 - 0.0279\lambda^6}.$$

The poles are 0.5520, $0.2750 \pm 0.4804j$, $-0.2763 \pm 0.4736j$, -0.5483.

The Bode diagram of the two components of the filter is shown in Fig. 7.20 and the result of some simulations obtained with sinusoidal input functions and parameters

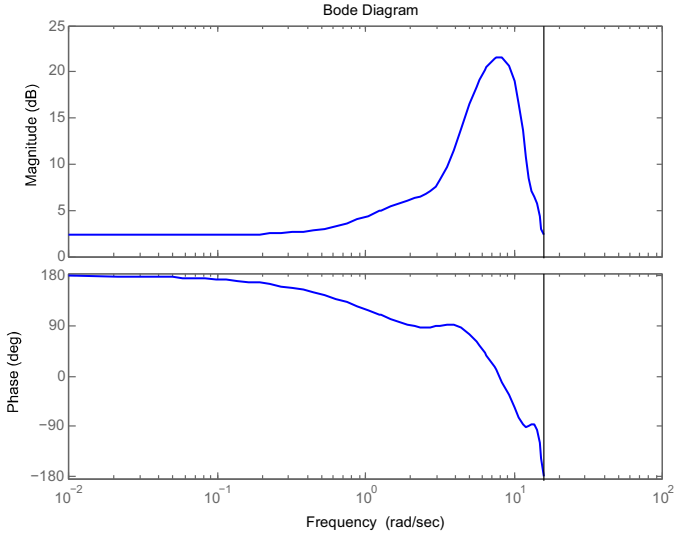
```
omega_max = 16;
ampiezza_max = 100; % maximum amplitude value of u
p_change_u = 0.5;
```

is shown in Fig. 7.21.

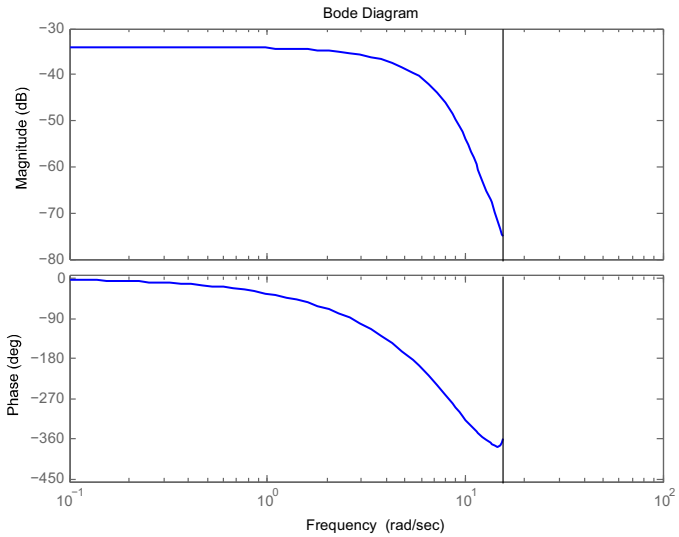
Remark

Again, in both cases, even though the estimation error is not always included in the interval $\pm\mu_{opt}$, the filter furnishes a good estimation, since $\hat{z}$ tracks properly z and the estimation error is small in proportion to the amplitude of the signal. The second example underlines that, in presence of a known input, the estimation error may grow as z (and therefore $\hat{z}$) grows.

[4] The other values of a_i are not exactly zero, but they are very close to zero

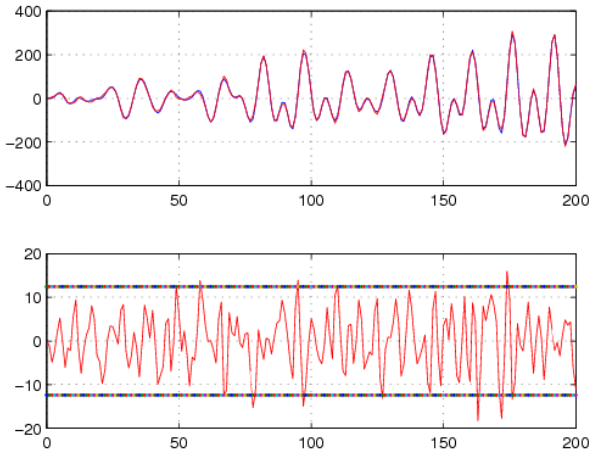


(a) $\frac{b(\lambda)}{a(\lambda)}$

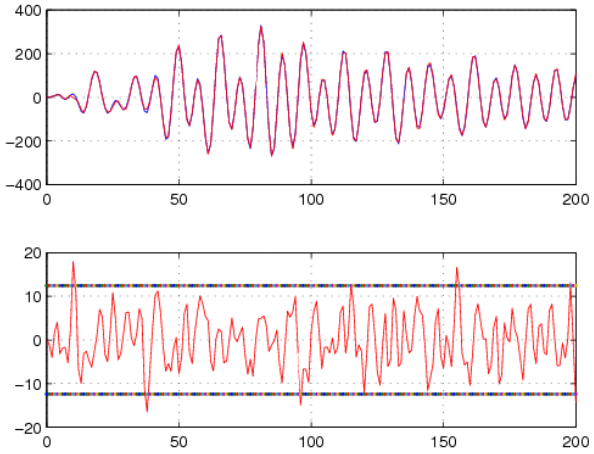


(b) $\frac{c(\lambda)}{a(\lambda)}$

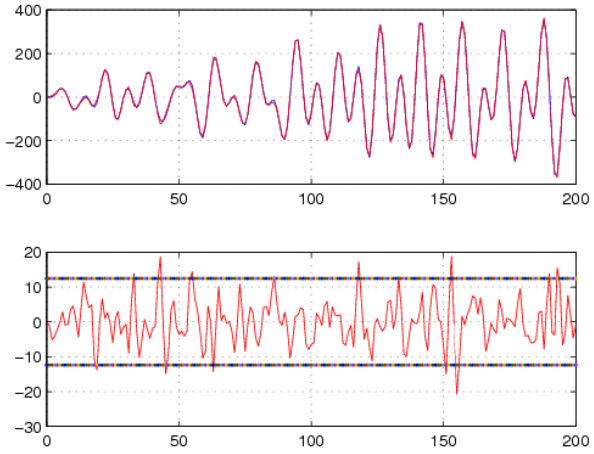
Figure 7.18: Bode diagram of the filters for the fourth order quantized system – first example



(a)

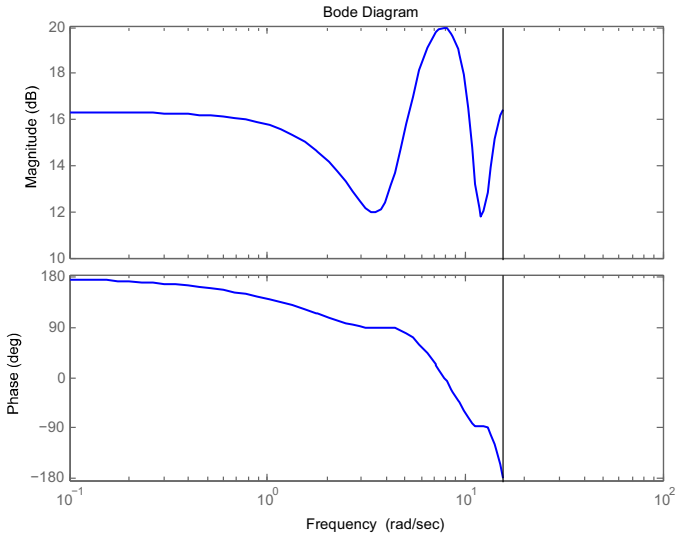


(b)

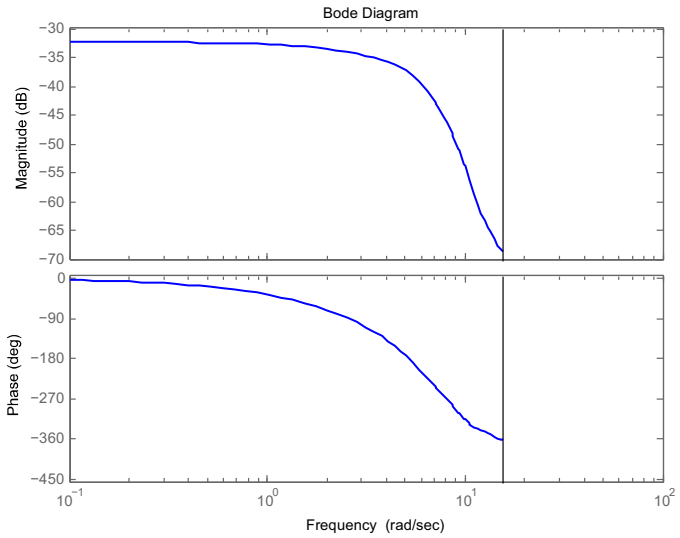


(c)

Figure 7.19: First set of simulations of the filter behaviour for fourth order quantized system with sinusoidal input function. Above z, in blue, is compared with $\hat{z}$, in red; below, the estimation error is shown and $\pm\mu_{opt}$ are marked

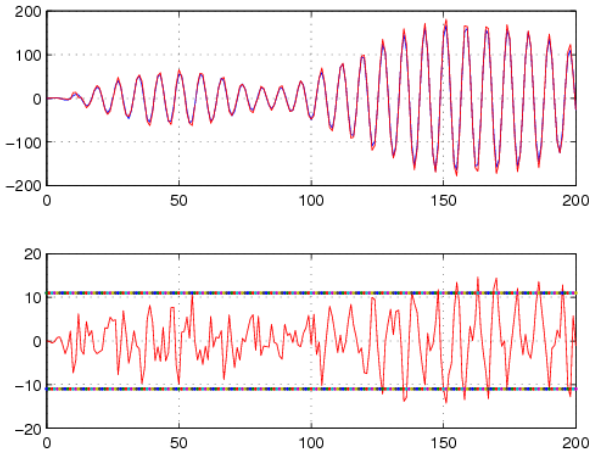


(a) $\frac{b(\lambda)}{a(\lambda)}$

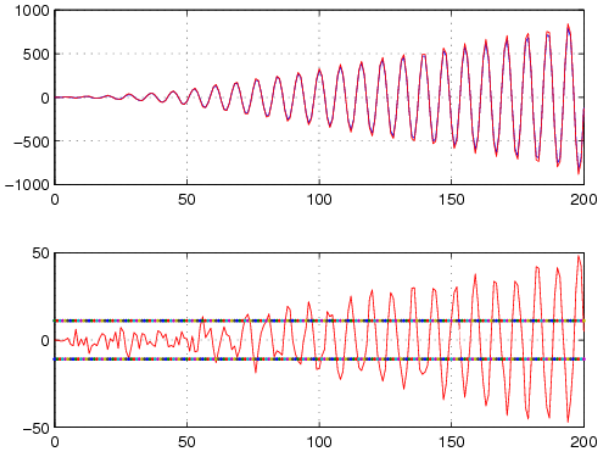


(b) $\frac{c(\lambda)}{a(\lambda)}$

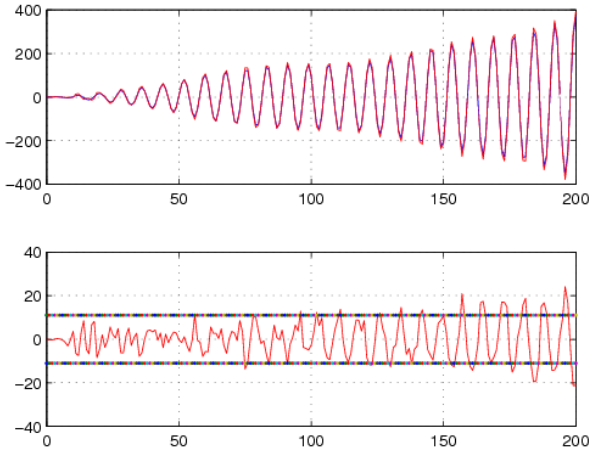
Figure 7.20: Bode diagram of the filters for the fourth order quantized system – second example



(a)



(b)



(c)

Figure 7.21: Second set of simulations of the filter behaviour for fourth order quantized system with sinusoidal input function. Above z, in blue, is compared with $\hat{z}$, in red; below, the estimation error is shown and $\pm\mu_{opt}$ are marked

7.3.4 Comparison with Kalman Filter

We consider $\omega_1 = 4$, $\omega_2 = 4$, with $T_s = 0.2$, $\beta = 1$, $\gamma = 1$.

The discrete–time Kalman filter transfer function is

$$\frac{b_{kalman}(\lambda)}{a_{kalman}(\lambda)} \approx \frac{-0.1893\lambda + 0.2484\lambda^2 - 0.1679\lambda^3 - 0.0105\lambda^4}{1 - 2.7382\lambda + 3.8052\lambda^2 - 2.6443\lambda^3 + 0.9326\lambda^4}$$

and its poles are $0.6725 \pm 0.7171j$, $0.6966 \pm 0.6926j$.

We compare this Kalman filter with the r–equalized, $r = 4$, which attains $\mu_{opt} \approx 19.2$:

$$\frac{b(\lambda)}{a(\lambda)} \approx \frac{6.0711 - 8.2638\lambda + 0.0070\lambda^2 + 2.2338\lambda^3 - 1.3430\lambda^4}{1 + 0.0000\lambda + 0.0007\lambda^2 - 0.0001\lambda^3 - 0.0633\lambda^4}$$

and

$$\frac{c(\lambda)}{a(\lambda)} \approx \frac{0 + 0.0009\lambda + 0.0024\lambda^2 + 0.0014\lambda^3 + 0.0002\lambda^4}{1 + 0.0000\lambda + 0.0007\lambda^2 - 0.0001\lambda^3 - 0.0633\lambda^4}.$$

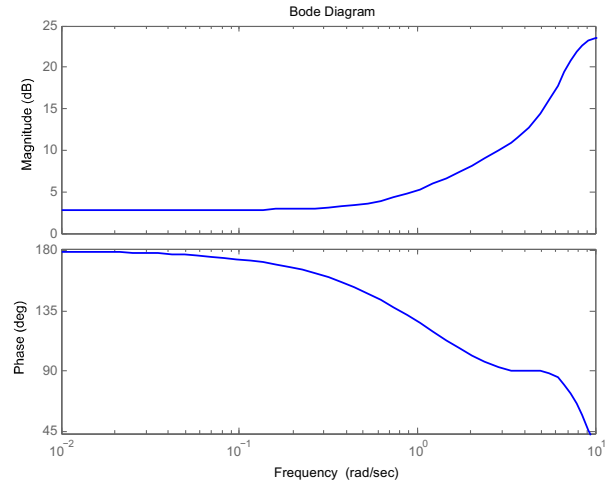
The poles are 0.5014, -0.5011, $-0.0002 \pm 0.5019j$.

The Bode diagrams are shown in Fig. 7.22 and the result of some comparative simulations obtained with sinusoidal input functions and parameters

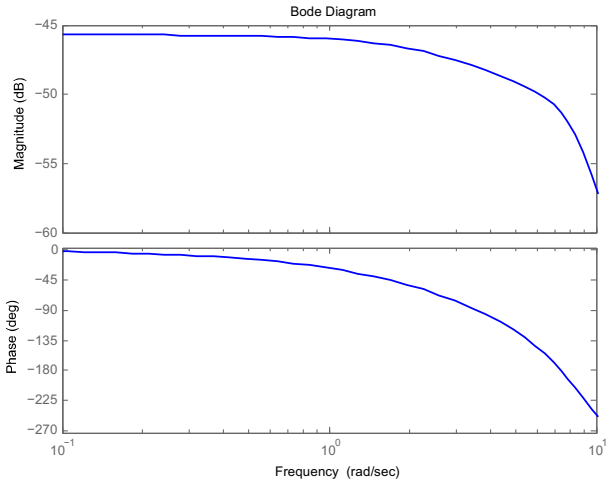
```
omega_max = 5;
ampiezza_max = 45; % maximum amplitude value of u
p_change_u = 0.2;
```

is shown in Fig. 7.23.

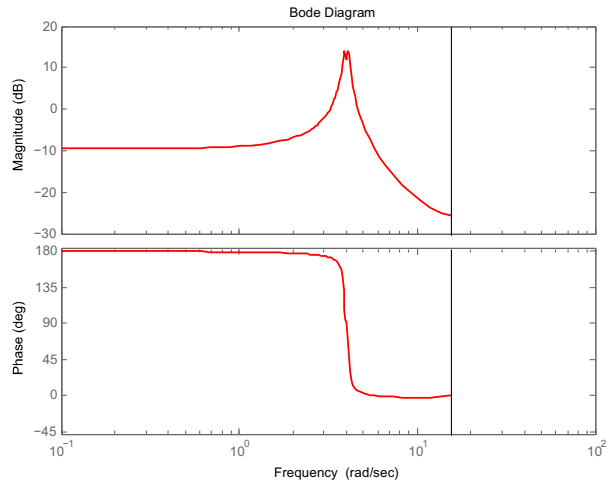
Both the filters behave well, but several experiments with randomly generated noise show that, in general, the equalized filter has a smaller worst case error.



(a) $\frac{b(\lambda)}{a(\lambda)}$

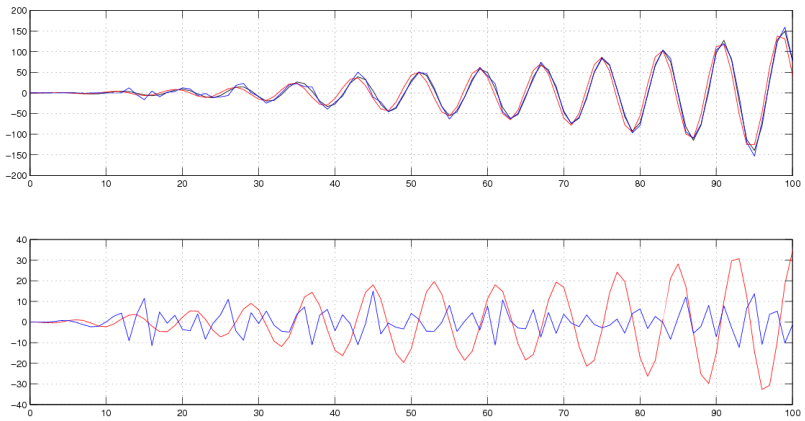


(b) $\frac{c(\lambda)}{a(\lambda)}$

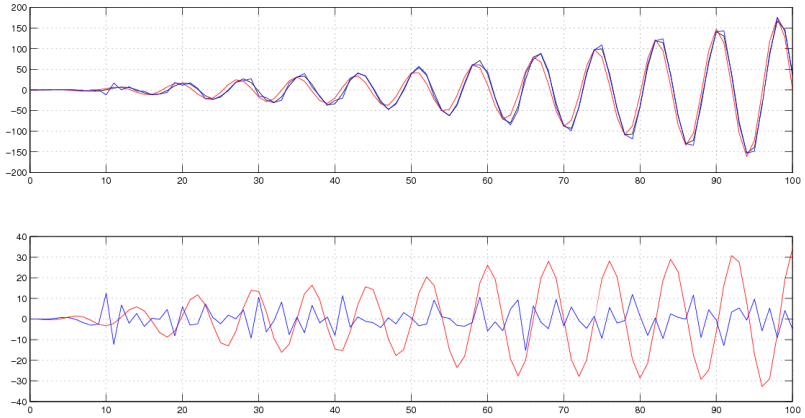


(c) $\frac{b_{kal}(\lambda)}{a_{kal}(\lambda)}$

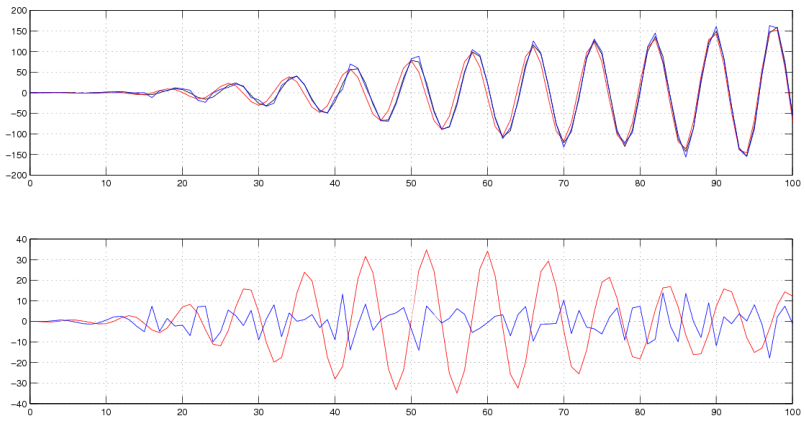
Figure 7.22: Bode diagram of the Kalman filter (in red) and of the equalized filter (in blue) for the fourth order quantized system



(a)



(b)



(c)

Figure 7.23: Simulations for fourth order quantized system with sinusoidal input function. Above z, in black, is compared with $\hat{z}_{eq}$, in blue, and $\hat{z}_{kalman}$, in red; below, the estimation error e_{eq}, in blue, is compared with e_{kalman}, in red

7.4 System with Selective Prefilters

7.4.1 Model Construction

Suppose that the second order system with continuous–time transfer function from input u to the output to estimate, z,

$$\frac{s}{s^2+\omega^2}$$

and continuous–time transfer function from input u to measured output y

$$\frac{1}{s^2+\omega^2},$$

is affected by disturbances having a dominant frequency. This can be obtained if the random noise v is modulated by a selective prefilter with transfer function

$$\frac{1}{s^2+\omega_v^2}$$

and then added to the input u. Likewise, measured output y contains a noise w which is modulated by a prefilter with transfer function

$$\frac{1}{s^2+\omega_w^2}.$$

The Bode diagram of such a selective prefilter is shown in Fig. 7.24: it is evident that it attenuates all frequencies except a particular range which is strongly amplified.

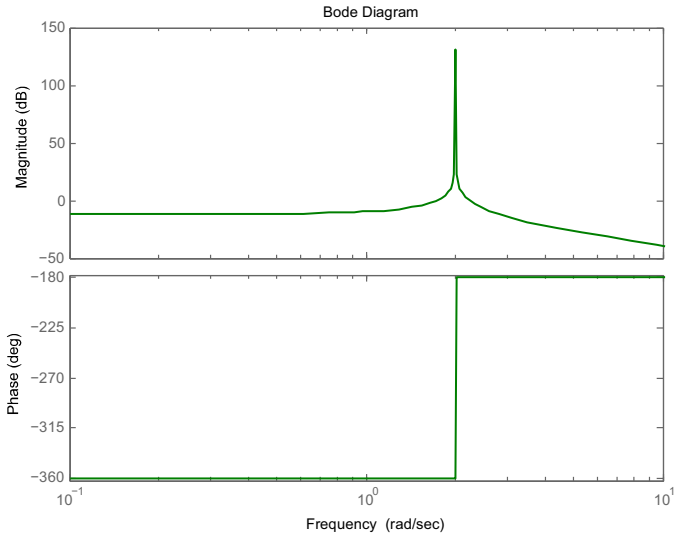


Figure 7.24: Selective prefilter frequency response: $\frac{1}{s^2+\omega^2}$ with $\omega = 2$

Fig. 7.25 shows the complete block scheme of the system with filters applied. Note that, after the disturbances have been selectively prefiltered, a saturation to $\pm\gamma$ and $\pm\beta$ is imposed respectively to v and to w, in order to avoid the noise exceeding its limit.

7.4.2 Filter Synthesis and Simulation

If $\omega = 4$, $\omega_v = 4$, $\omega_w = 2$, $T_s = 0.2$, $\beta = 1$, $\gamma = 1$, $r = 3$, we obtain $\mu_{opt} \approx 4.4$; the two components of the filter are

$$\frac{b(\lambda)}{a(\lambda)} \approx \frac{-0.0000 - 0.1628\lambda - 3.9105\lambda^2 + 0.0000\lambda^3}{1 - 0.0000\lambda + 0.0000\lambda^2 + 0.0002\lambda^3}$$

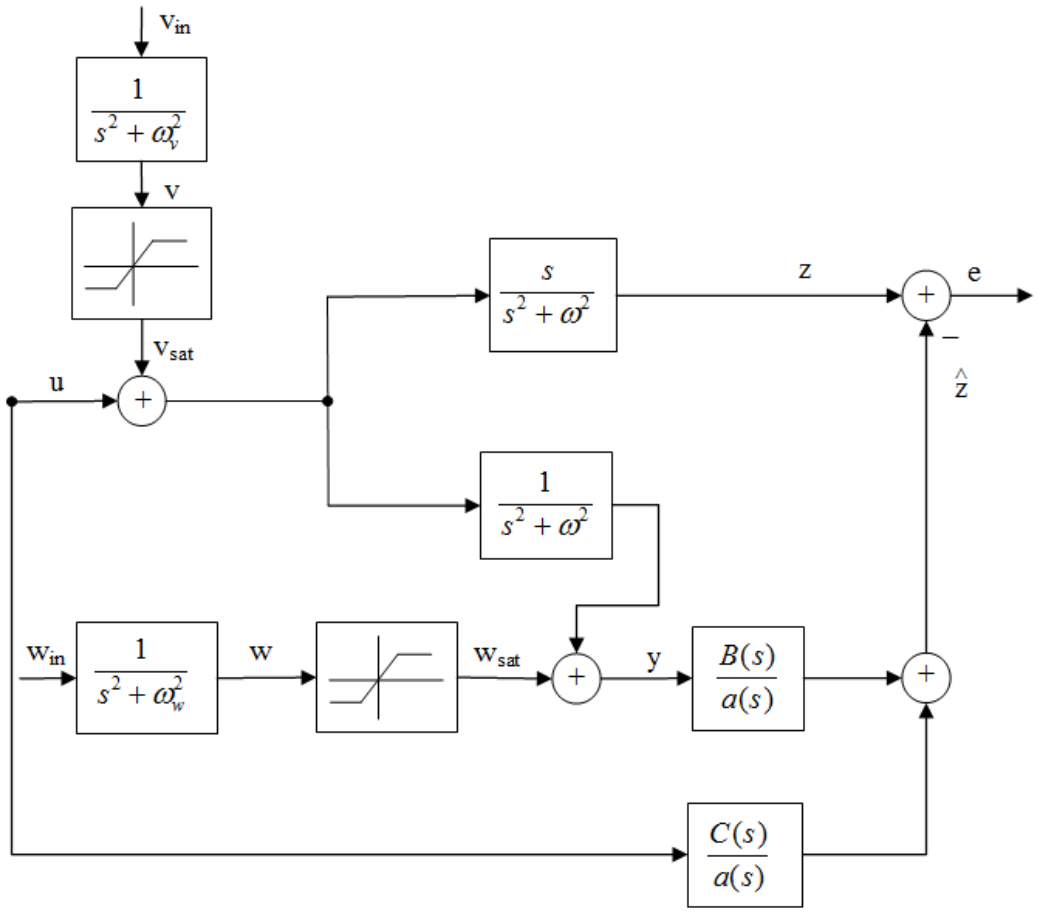


Figure 7.25: Filtering scheme of the system with prefilters

and

$$\frac{c(\lambda)}{a(\lambda)} \approx \frac{0 + 0.1793\lambda + 0.0716\lambda^2 - 0.0000\lambda^3}{1 - 0.0000\lambda + 0.0000\lambda^2 + 0.0002\lambda^3}\,(^5).$$

The poles are $0.0269 \pm 0.0466j$, -0.0538.

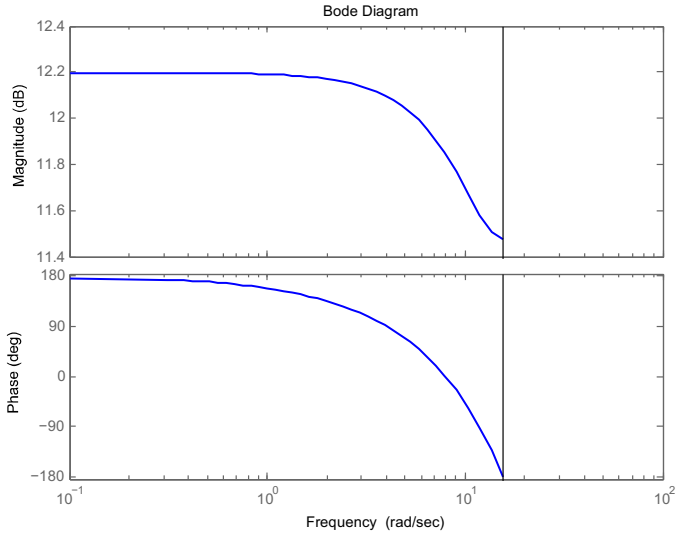
The Bode diagram of the two components of the filter is shown in Fig. 7.26 and the result of some simulations obtained with sinusoidal input functions and parameters

```
omega_max = 3;
omega_max_u = 4; % maximum amplitude value of u
ampiezza_max = 20;
p_change = 0.5;
p_change_u = 0.2;
```

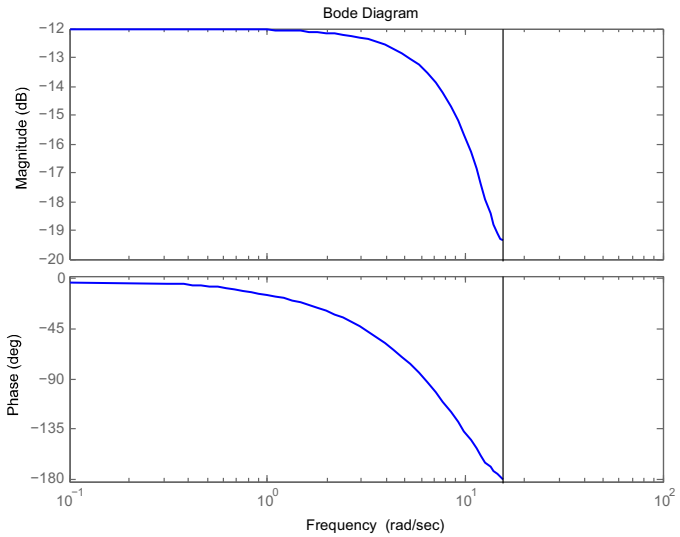
is shown in Fig. 7.27.

The filter works well, because $\hat{z}$ reproduces z with an estimation error which is much smaller than the amplitude of the signal, even though it is not always contained in the interval $\pm\mu_{opt}$.

[5] All 0.0000 values in a, b and c are not exactly zero, but they are very close to zero

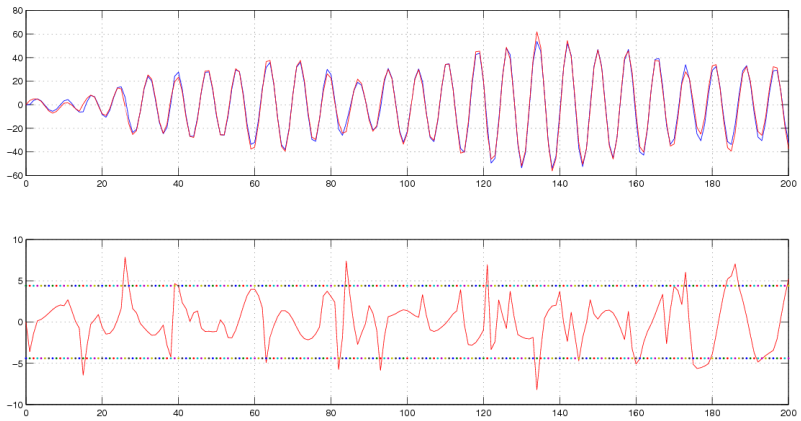


(a) $\frac{b(\lambda)}{a(\lambda)}$

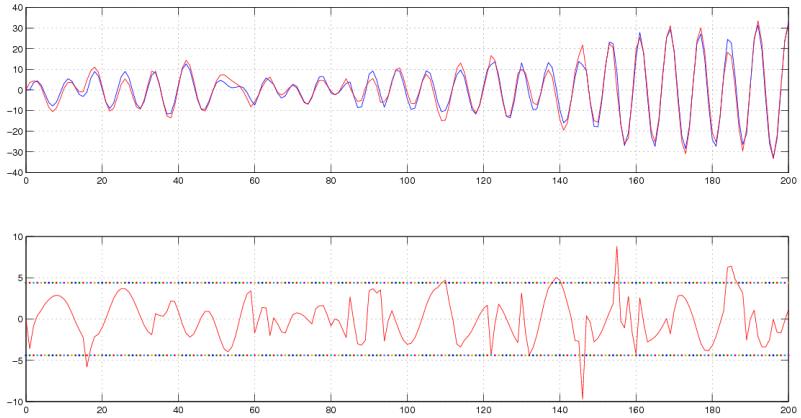


(b) $\frac{c(\lambda)}{a(\lambda)}$

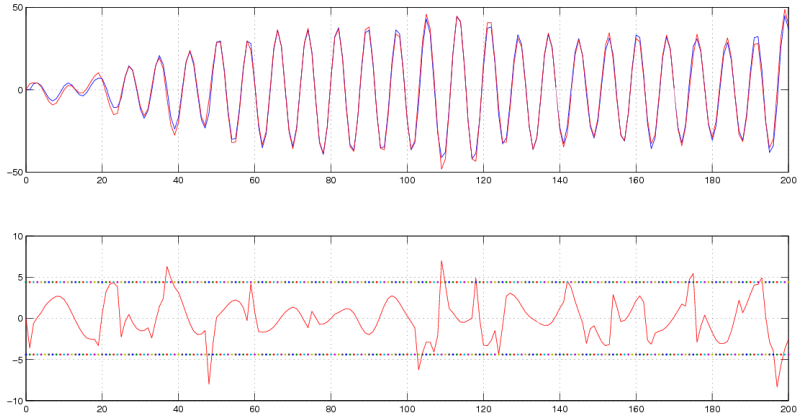
Figure 7.26: Bode diagram of the filters for the system with prefilters



(a)



(b)



(c)

Figure 7.27: Simulations of the filter behaviour for the system with prefilters (sinusoidal input function). Above z, in blue, is compared with $\hat{z}$, in red; below, the estimation error is shown and $\pm\mu_{opt}$ are marked

7.4.3 Comparison with Kalman Filter

The discrete–time Kalman filter transfer function is

$$\frac{b_{kalman}(\lambda)}{a_{kalman}(\lambda)} \approx \frac{-0.1325\lambda - 0.0055\lambda^2}{1 - 1.3593\lambda + 0.9525\lambda^2}$$

and its poles are $0.6797 \pm 0.7004j$.

We compare this Kalman filter with the r–equalized, $r = 2$, which attains $\mu_{opt} \approx 4.4$:

$$\frac{b(\lambda)}{a(\lambda)} \approx \frac{-0.0000 - 0.1629\lambda - 3.8854\lambda^2}{1}$$

and

$$\frac{c(\lambda)}{a(\lambda)} \approx \frac{0 + 0.1793\lambda + 0.0737\lambda^2}{1}.$$

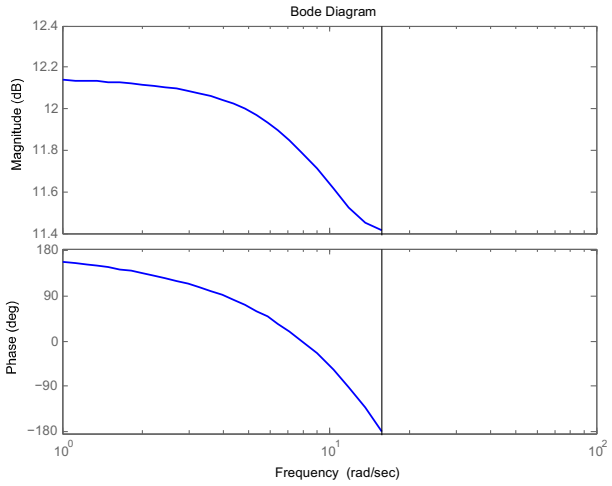
The two components of the filter are FIR.

The Bode diagrams are shown in Fig. 7.28 and the result of some comparative simulations obtained with sinusoidal input functions and parameters

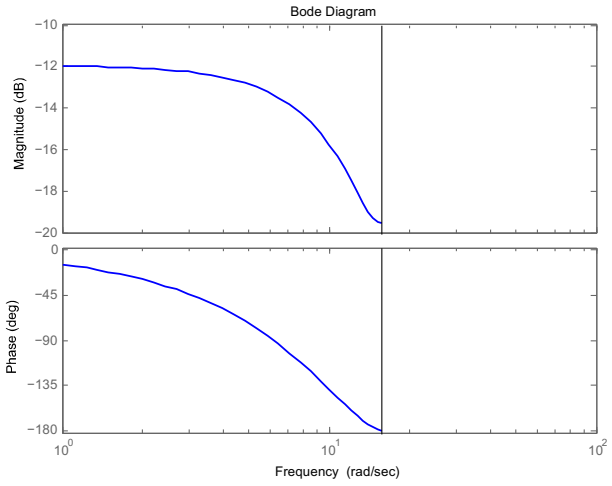
```
omega_max = 3;
omega_max_u = 4; % maximum amplitude value of u
ampiezza_max = 20;
p_change = 0.5;
p_change_u = 0.2;
```

is shown in Fig. 7.29.

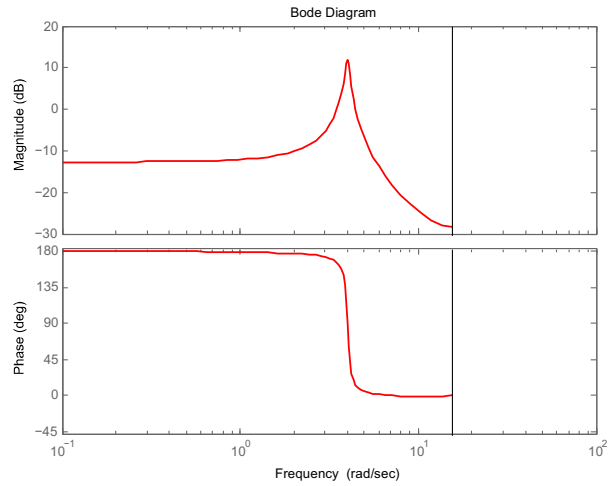
Both the filters behave well, but several experiments with randomly generated disturbances show that the equalized filter usually has a smaller worst case error.



(a) $\frac{b(\lambda)}{a(\lambda)}$

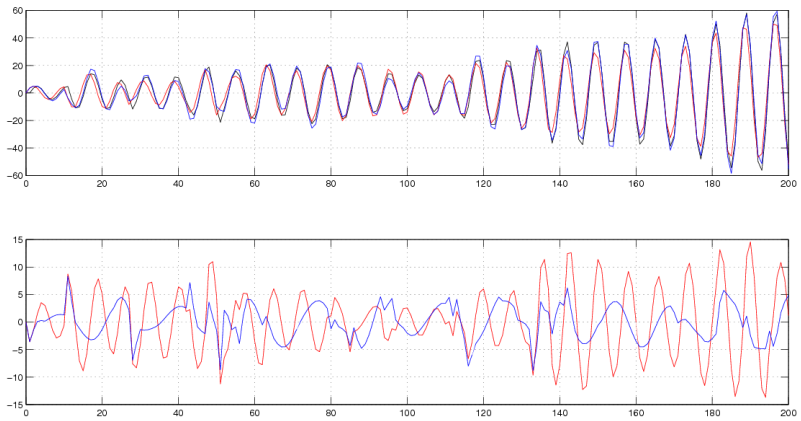


(b) $\frac{c(\lambda)}{a(\lambda)}$

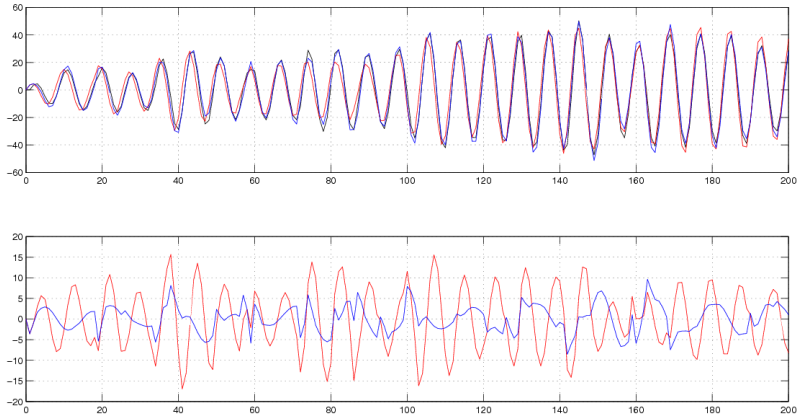


(c) $\frac{b_{kal}(\lambda)}{a_{kal}(\lambda)}$

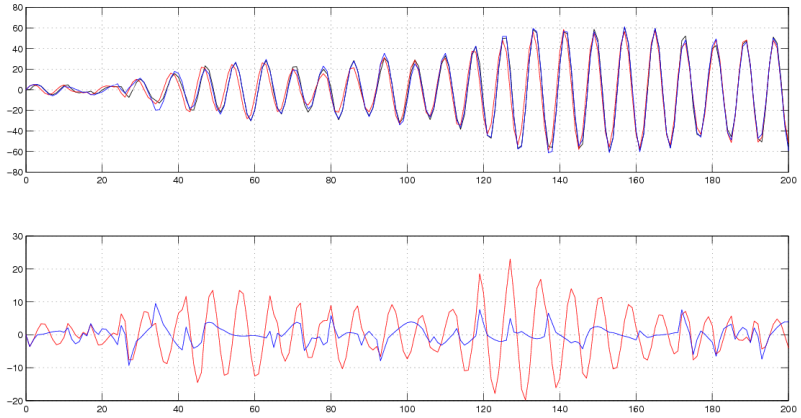
Figure 7.28: Bode diagram of the Kalman filter (in red) and of the equalized filter (in blue) for the system with selective prefilters



(a)



(b)



(c)

Figure 7.29: Simulations for the system with selective prefilters, with sinusoidal input function. Above z, in black, is compared with $\hat{z}_{eq}$, in blue, and $\hat{z}_{kalman}$, in red; below, the estimation error e_{eq}, in blue, is compared with e_{kalman}, in red

Chapter 8

Conclusions

Filtering in presence of unknown, but bounded, disturbances aims at confining the estimation error within a bounded set. The new approach here considered (and presented in [1]) is based on the concept of *equalized performance*: it works with r–length estimation error sequences and allows to confine them to the tightest possible hyperrectangle. Its main advantages are that:

- the objective is accomplished with an r^{th} order LTI filter, whose coefficients can be *easily* found via a convex optimization algorithm;
- the multiple output case can be *easily* faced by considering a set of scalar filters;
- the MATLAB simulations implemented show a *good behaviour* of the filters obtained, which are comparable to the Kalman filter, and often rather better.

The application of filters synthesized with this new approach has been extended to systems not only affected by noise, but also with the additional presence of an external, known (control) input coming from the same channel of the process noise. Several MATLAB simulations have shown that these filters behave well and are comparable to the Kalman filter, and often better.

Until now, only LTI systems have been considered for synthesizing filters with this new procedure. Current research is seeking to extend this approach to filter synthesis also to switched and piecewise linear systems.

I heartily thank Prof. Blanchini
for his valuable advice, his continuous care and his kindness.

Bibliography

[1] **F. Blanchini and M. Sznaier, "A Convex Optimization Approach to Synthesizing Bounded Complexity ℓ^∞ Filters", Proc. 48$^{\text{th}}$ IEEE Conference on Decision and Control, held jointly with the 28$^{\text{th}}$ Chinese Control Conference, Shanghai, China, pp. 217–222, 15–18 Dec. 2009.**

[2] F. Blanchini and M. Sznaier, "A Convex Optimization Approach to Fixed–Order Controller Design for Disturbance Rejection for SISO Systems", IEEE Transactions on Automatic Control, Vol. AC–45, n. 4, pp. 784–789, April 2000.

[3] B. T. Polyak and M. E. Halpern, "The Use of a New Optimization Criterion for Discrete–Time Feedback Controller Design", Proc. 38$^{\text{th}}$ IEEE CDC, Tucson, AZ, pp. 894–899, Dec. 1999.

[4] D. Luenberger, "An Introduction to Observers", IEEE Trans. Automat. Control, Vol. 16, n. 6, pp. 596–602, 1971.

[5] C. Tsui, "A New Design Approach to Unknown Input Observers", IEEE Trans. Automat. Control, Vol. 41, pp. 464–468, 1996.

[6] E. Fornasini and G. Marchesini, "Appunti di Teoria dei Sistemi", Padova, Edizioni Libreria Progetto, 2003.

[7] M. Grant and S. Boyd, "CVX: Matlab software for disciplined convex programming", version 1.21, http://cvxr.com/cvx, May 2010.

[8] S. Boyd and L. Vandenberghe, "Convex Optimization", Cambridge University Press, 2004.

[9] F. C. Schweppe, "Recursive State Estimation: Unknown but Bounded Errors and System Inputs", IEEE Trans. Automat. Contr., Vol. 13, pp. 22–28, 1968.

[10] D. P. Bertsekas and I. B. Rhodes, "Recursive State Estimation for a Set-Membership Description of Uncertainty", IEEE Trans. Automat Control, Vol. 16, pp. 117–128, 1971.

[11] M. Milanese and G. Belforte, "Estimation Theory and Uncertainty Intervals Evaluation in Presence of Unknown but Bounded Errors: Linear Families of Models and Estimators", IEEE Trans. Automat Control, Vol. 27, n. 2, pp. 408–414, April 1982.

[12] M. Milanese and R. Tempo, "Optimal Algorithms Theory for Robust Estimation and Prediction", IEEE Trans. Automat Control, Vol. 30, n. 5, pp. 730–738, 1985.

[13] R. Tempo, "Robust Estimation and Filtering in the Presence of Bounded Noise", IEEE Trans. Aut. Control, Vol. 33, n. 9, pp. 864–867, 1988.

[14] M. Milanese and A. Vicino, "Optimal Estimation Theory for Dynamic Systems with Set Membership Uncertainty: An Overview", Automatica, Vol. 27, pp. 997–1009, 1991.

[15] A. A. Stoorvogel, "ℓ^1 State Estimation for Linear Systems Using Non–Linear Observers", Proc. 35$^{\text{th}}$ IEEE CDC, Kobe, Japan, pp. 2407–2411, Dec. 1996.

[16] P. Voulgaris, "On Optimal ℓ^∞ to ℓ^∞ Filtering", Automatica, Vol. 31, pp. 489–495, 1995.

[17] N. E. Barabanov and O. N. Granichin, "Optimal Controller for a Linear Plant with Bounded Noise", Autom. Remote Control, Vol. 45, n. 5, pp. 578–584, 1984.

[18] M. A. Dahleh and I. D. Diaz-Bobillo, "Control of Uncertain Systems: A Linear Programming Approach", Prentice-Hall, Englewood Cliffs, New Jersey, USA, 1995.

[19] M. A. Dahleh and J. B. Pearson, "ℓ^1–Optimal Feedback Controllers for MIMO Discrete–Time System", IEEE Trans. Automat Control, Vol. 32, n. 4, pp. 314–322, 1987.

[20] J. S. Shamma and K. Y. Tu, "Approximate Set–Valued Observers for Nonlinear Systems", IEEE Trans. Automat Control, Vol. 42, n. 5, pp. 648–658, 1997.

[21] J. S. Shamma and K. Y. Tu, "Set–Valued Observer and Optimal Disturbance Rejection", IEEE Trans. Automat. Control, Vol. 44, n. 2, pp. 253–164, Feb. 1999.

Part III

Appendices

Appendix A

Source Code of MATLAB Programs

A.1 MATLAB Code Usage

To obtain the simulations for every example, several MATLAB files have been written:

- a `Data_File`: it contains all the parameters for filter synthesis and simulation that can be modified by the user
- a `Filter_Synthesis` file: it generates the equalized filter as described in [1] and uses the routine *bos*
- a `Simulation` file: it generates the simulation diagrams and graphs

To perform simulations that show the equalized filter behaviour:

0. Set the desired values in `Data_File`
1. Run `Data_File`
2. Run `Filter_Synthesis`
3. If the value of `mrgn` is positive and as small as possible, go to step 4. If it is not, go back to step 0. and modify the value of `mu`: increase it if `mrgn` is negative; decrease it if `mrgn` is positive
4. Run `Simulation`

The Bode diagram of the filter is shown; then, in the same figure, the graph that compares z with its estimate $\hat{z}$ and the graph of the estimation error are shown.

In order to compare the equalized with the Kalman filter, a `Kalman` file has been written to synthesize the Kalman filter and a `Comparative_Simulation` file to confront the behaviour of the two filters. To perform simulations:

0. Set the desired values in `Data_File_Comparison`
1. Run `Data_File_Comparison`
2. Run `Filter_Synthesis`
3. If the value of `mrgn` is positive and as small as possible, go to step 4. If it is not, go back to step 0. and modify the value of `mu`: increase it if `mrgn` is negative; decrease it if `mrgn` is positive
4. Run `Kalman`

5. Run `Comparative_Simulation`

The Bode diagrams of the two filters are shown; then, in the same figure, the graph that compares z with its two different estimates and the graph of the two estimation errors are shown.

All the MATLAB code used for the simulations is enclosed with the thesis in a CD; here just two examples of usage are reported.

A.2 *bos* function by Blanchini and Sznaier

```
function [a,b,c,mrgn] = bos(d,m,n,beta,gamma,ord_com,mu)
[ord_sys,dummy]=size(d);

% the first column is divided by mu but it should not
M=convmtx(m,ord_com+1)/mu;
N=convmtx(n,ord_com+1)/beta;
D=convmtx(d,ord_com+1)/gamma;
Phi =[M -N -D]
[n_rig,n_col]=size(Phi);
P=Phi(:,2:n_col);
n_col=n_col-1;
% this is to restore the first column correct value
q=-Phi(:,1)*mu;

P
q

cvx_begin
variable x(n_col)
minimize norm(x,1)
subject to
P*x==q
cvx_end

mrgn=mu-norm(x,1);
a=[1 ; x(1:ord_com)/mu];
b=x(ord_com+1:2*ord_com+1)/beta;
c=x(2*ord_com+2:3*ord_com+2)/gamma;
check_eq_zero=norm(conv(m,a)-conv(b,n)-conv(c,d))

dir_gain=sum(m)/sum(d)
ext_gain=(sum(n)/sum(d))*(sum(b)/sum(a))
```

A.3 Example: a Car Suspension Model

Here we report the code for the simulation of the equalized filter behaviour, for a system in presence of disturbances only.

Data File Examples

Sinusoidal Input Function

```
% SYSTEM PARAMETERS
k1 = 2;
k2 = 3;
h = 0.1;
```

```
Ts = 1;     % Sampling rate for discretization

% TO OBTAIN OPTIMAL FILTER
beta = 1;
gamma = 10;
ord_com = 6;
mu = 18.7;

% TO RUN SIMULATION
% Maximum frequency value of sinusoidal disturbances v and w
omega_max = 50;
% Probability to change the frequency value of disturbances v and w
p_change_v = 0.2;
p_change_w = 0.5;
% Simulation length
N_step = 50;
```

Linear Input Function

```
% SYSTEM PARAMETERS
k1 = 1;
k2 = 1;
h = 0.1;

Ts = 1;     % Sampling rate for discretization

% TO OBTAIN OPTIMAL FILTER
beta = 1;
gamma = 10;
ord_com = 6;
mu = 17.3;

% TO RUN SIMULATION
% Minimum and maximum angular coefficient for disturbance v
alfa_min = -2;
alfa_max = 2;
% Minimum and maximum angular coefficient for disturbance w
ang_min = -1;
ang_max = 1;
% Probability to change the angular coefficient of disturbances v and w
p_change_v = 0.2;
p_change_w = 0.5;
% Simulation length
N_step = 50;
```

Filter Synthesis

```
% Continuous-time
F=[0 0 1 0; 0 0 0 1; -(k1+k2) k2 -h h; k2 -k2 h -h];
G=[0; 0; k1; 0];
Hz=[1 0 0 0];
Hy=[1 -1 0 0];

sysc_z = ss(F, G, Hz, 0);
sysc_y = ss(F, G, Hy, 0);
```

```
% Discrete-time
sysd_z = c2d(sysc_z, Ts);
sysd_y = c2d(sysc_y, Ts);

[Azd,Bzd,Czd,Dzd] = ssdata(sysd_z);
[Ayd,Byd,Cyd,Dyd] = ssdata(sysd_y);

[m, d] = ss2tf (Azd,Bzd,Czd,Dzd);
[n, d] = ss2tf (Ayd,Byd,Cyd,Dyd);

m=m';
n=n';
d=d';

[a,b,c,mrgn] = bos(d,m,n,beta,gamma,ord_com,mu)

a=a';
b=b';
c=c';
```

Simulation

Sinusoidal Input Function

```
[m, d] = ss2tf (Azd,Bzd,Czd,Dzd);
[n, d] = ss2tf (Ayd,Byd,Cyd,Dyd);

[dim, dim]=size(Ayd);

% State initialization
x = zeros(dim,1);
x_eq = zeros(ord_com,1);

% Systems
[A,B,Cz,Dz] = tf2ss(m,d);
[A,B,Cy,Dy] = tf2ss(n,d);
[A_eq,B_eq,C_eq,D_eq] = tf2ss(b,a);

% Bode diagram of the filter
figure(1)
dbode(b,a,Ts)

z_plot = 0;
z_eq_plot = 0;
t_plot = 0;
y = 0;
z = 0;

% Start simulation
omega_v = rand*omega_max;
omega_w = rand*omega_max;

for k=0:N_step

    change_v = rand;
    if change_v < p_change_v
```

```
        omega_v = rand*omega_max;
    end
    v=gamma*cos(omega_v*k*Ts);

    change_w = rand;
    if change_w < p_change_w
        omega_w = rand*omega_max;
    end
    w=beta*cos(omega_w*k*Ts);

    % Equalized filter
    z_eq = C_eq *x_eq + D_eq * y;
    x_eq = A_eq *x_eq + B_eq * y;

    % Plot
    t_plot = [t_plot k];
    z_plot = [z_plot z];
    z_eq_plot = [z_eq_plot z_eq];

    % Plant
    z=Cz*x+Dz*v;
    y=Cy*x+Dy*v;
    x=A*x+B*v;
    y=y+w;
end
figure(2)
 subplot(2,1,1)
 plot(t_plot,z_plot,'b',t_plot,z_eq_plot,'r')
 grid
 subplot(2,1,2)
 plot(t_plot,z_plot-z_eq_plot,'r', t_plot, mu, '.-', t_plot, -mu, '.-')
 grid
```

Linear Input Function

```
[m, d] = ss2tf (Azd,Bzd,Czd,Dzd);
[n, d] = ss2tf (Ayd,Byd,Cyd,Dyd);

[dim, dim]=size(Ayd);

% State initialization
x = zeros(dim,1);
x_eq = zeros(ord_com,1);

% Systems
[A,B,Cz,Dz] = tf2ss(m,d);
[A,B,Cy,Dy] = tf2ss(n,d);
[A_eq,B_eq,C_eq,D_eq] = tf2ss(b,a);

% Bode diagram of the filter
figure(1)
dbode(b,a,Ts)

z_plot = 0;
z_eq_plot = 0;
t_plot = 0;
```

```
y = 0;
z = 0;

% Start simulation
v = 0;
w = 0;
r = rand;
alfa = alfa_min*r+alfa_max*(1-r);
h = rand;
ang = ang_min*h+ang_max*(1-h);

for k=0:N_step

    change_v = rand;
    if change_v < p_change_v
        r = rand;
        alfa = alfa_min*r+alfa_max*(1-r);
    end
    v = v+alfa*Ts;
    if v > gamma
       v = gamma;
    elseif v < -gamma
       v = -gamma;
    end

    change_w = rand;
    if change_w < p_change_w
        h = rand;
        ang = ang_min*h+ang_max*(1-h);
    end
    w = w+ang*Ts;
    if w > beta
        w = beta;
    elseif w < -beta
        w = -beta;
    end

    % Equalized filter
    z_eq = C_eq *x_eq + D_eq * y;
    x_eq = A_eq *x_eq + B_eq * y;

    % Plot
    t_plot = [t_plot k];
    z_plot = [z_plot z];
    z_eq_plot = [z_eq_plot z_eq];

    % Plant
    z=Cz*x+Dz*v;
    y=Cy*x+Dy*v;
    x=A*x+B*v;
    y=y+w;
end
figure(2)
 subplot(2,1,1)
 plot(t_plot,z_plot,'b',t_plot,z_eq_plot,'r')
 grid
```

```
subplot(2,1,2)
plot(t_plot,z_plot-z_eq_plot,'r', t_plot, mu, '.-', t_plot, -mu, '.-')
grid
```

A.4 Example: Fourth Order Quantized System

Here we report the code for the comparative simulation between the equalized and the Kalman filter, for a system in presence of a known (control) input too.

Data File Example

```
% SYSTEM PARAMETERS
omega1 = 4;
omega2 = 4;
Ts = 0.2;          % Sampling time for discretization

% TO OBTAIN OPTIMAL FILTER
beta = 1;
gamma = 1;
ord_com = 4;
mu = 19.2;

% TO RUN SIMULATION
% Maximum frequency value and amplitude value of sinusoidal input u
omega_max = 5;
ampiezza_max = 45;
% Probability to change the frequency value of input u
p_change_u = 0.2;
% Simulation length
N_step = 100;
```

Filter Synthesis

```
sysc_z = tf ( [0 1 0] , conv([1 0 omega1^2],[1 0 omega2^2]) );
sysc_y = tf ( [0 0 1] , conv([1 0 omega1^2],[1 0 omega2^2]) );

sysd_z = c2d(sysc_z, Ts);
sysd_y = c2d(sysc_y, Ts);

[Azd,Bzd,Czd,Dzd] = ssdata(sysd_z);
[Ayd,Byd,Cyd,Dyd] = ssdata(sysd_y);

[m, d] = ss2tf (Azd,Bzd,Czd,Dzd);
[n, d] = ss2tf (Ayd,Byd,Cyd,Dyd);

m=m';
n=n';
d=d';

[a,b,c,mrgn] = bos(d,m,n,beta,gamma,ord_com,mu)

a=a';
b=b';
c=c';
```

Kalman Filter Synthesis

```
[m, d] = ss2tf (Azd,Bzd,Czd,Dzd);
[n, d] = ss2tf (Ayd,Byd,Cyd,Dyd);

[A,B,Cz,Dz] = tf2ss(m,d);
[A,B,Cy,Dy] = tf2ss(n,d);

[P,CLEV,G] = dare(Ayd',Cyd',Byd*Byd');

L=Ayd*P*Cyd'*(Cyd*P*Cyd'+1)^(-1);

A_kal = Ayd - L*Cyd;
B_kal = L;
C_kal = Czd;
D_kal = Dzd;
```

Comparative Simulation

```
[m, d] = ss2tf (Azd,Bzd,Czd,Dzd);
[n, d] = ss2tf (Ayd,Byd,Cyd,Dyd);

[dim, dim]=size(Ayd);

% State initialization
x = zeros(dim,1);
x_kal = zeros(dim,1);
x_b = zeros(ord_com,1);
x_c = zeros(ord_com,1);

% Systems
[A,B,Cz,Dz] = tf2ss(m,d);
[A,B,Cy,Dy] = tf2ss(n,d);
[A_b,B_b,C_b,D_b] = tf2ss(b,a);
[A_c,B_c,C_c,D_c] = tf2ss(c,a);
[b_kal,a_kal] = ss2tf(A_kal,B_kal,C_kal,D_kal);

% Bode diagram of the filter
figure(1)
dbode(b_kal,a_kal,Ts,'r')
figure(2)
dbode(b,a,Ts, 'b')
figure(3)
dbode(c,a,Ts, 'b')

z_plot = 0;
z_kal_plot=0;
z_eq_plot = 0;
t_plot = 0;
y = 0;
z = 0;

% Start simulation
r = rand*omega_max;

for k=0:N_step
```

```
    change = rand;
    if change < p_change_u
        r = rand*omega_max;
    end

    u = ampiezza_max*cos(r*k*Ts);
    u_q = round(u/(gamma*2))*gamma*2;

    % Filter - part b
    z_b = C_b *x_b + D_b * y;
    x_b = A_b *x_b + B_b * y;

    % Filter - part c
    z_c = C_c *x_c + D_c * u;
    x_c = A_c *x_c + B_c * u;

    % Total filter
    z_eq = z_b + z_c;

    % Kalman filter
    z_kal = C_kal *x_kal + D_kal * y;
    x_kal = A_kal *x_kal + B_kal * y + B * u_q;

    % Plot
    t_plot = [t_plot k];
    z_plot = [z_plot z];
    z_eq_plot = [z_eq_plot z_eq];
    z_kal_plot = [z_kal_plot z_kal];

    % Plant
    z=Cz*x+Dz*u_q;
    y_ex=Cy*x+Dy*u_q;
    x=A*x+B*u_q;
    y = round(y_ex/(beta*2))*beta*2;

end
figure(4)
 subplot(2,1,1)
 plot(t_plot,z_plot,'k',t_plot,z_kal_plot,'r',t_plot,z_eq_plot,'b')
 grid
 subplot(2,1,2)
 plot(t_plot,z_plot-z_kal_plot,'r',t_plot,z_plot-z_eq_plot,'b')
 grid
```

Appendix B

Hints about Convex Optimization

Convex optimization is a subfield of mathematical optimization which studies the problem of minimizing convex functions. This special class of mathematical optimization problems includes least–squares and linear programming problems. Least–squares and linear programming problems have a fairly complete theory, arise in a variety of applications, and can be solved numerically very efficiently: the same can be said for the larger class of convex optimization problems. The convexity of the functions and of their domains makes the powerful tools of convex analysis applicable: this leads to a theory of necessary and sufficient conditions for optimality, a duality theory generalizing that for linear programming and effective computational methods.

We present here a summary of the basic concepts from [8] and many excerpts are quoted from the book. Just some hints are given about the wide and complex topic: for further deepening, consult [8].

Mathematical Optimization

A mathematical *optimization problem* has the form

$$\begin{array}{l} \text{minimize } f_0(x) \\ \text{subject to: } f_i(x) \leq b_i,\ i = 1, \ldots, m \end{array} \tag{B.1}$$

where the vector $x = (x_1, \ldots, x_n)$ is the *optimization variable* of the problem, the function $f_0 : \mathbb{R}^n \to \mathbb{R}$ is the *objective function*, the functions $f_i : \mathbb{R}^n \to \mathbb{R},\ i = 1, \ldots, m$ are the (inequality) *constraint functions*, and the constants $b_1, \ldots, b_m$ are the limits, or *bounds*, for the constraints. A vector x^* is called *optimal*, or a *solution* of the problem (B.1), if it has the smallest objective value among all vectors that satisfy the constraints: for any z with $f_1(z) \leq b_1, \ldots, f_m(z) \leq b_m$, we have $f_0(z) \geq f_0(x^*)$.

Optimization problems can be classified in families characterized by particular forms of the objective and constraint functions.

The optimization problem (B.1) is called a *linear program* if the objective and constraint functions $f_0, \ldots, f_m$ are linear (satisfy $f_i(\alpha x + \beta y) = \alpha f_i(x) + \beta f_i(y)$ for all $x, y \in \mathbb{R}^n$ and all $\alpha, \beta \in \mathbb{R}$).

In a convex optimization problem, the objective and constraint functions are convex: they satisfy the inequality

$$f_i(\alpha x + \beta y) \leq \alpha f_i(x) + \beta f_i(y) \tag{B.2}$$

for all $x, y \in \mathbb{R}^n$ and all $\alpha, \beta \in \mathbb{R}$ with $\alpha + \beta = 1,\ \alpha \geq 0, \beta \geq 0$. Convexity is more general than linearity: inequality replaces the more restrictive equality, and the inequality must hold only for certain values of α and β. Since any linear program is therefore a convex optimization problem, convex optimization can be considered a generalization of linear programming.

Affine and Convex Sets

Suppose $x_1 \neq x_2$ are two points in $\mathbb{R}^n$. Points of the form $y = \vartheta x_1 + (1-\vartheta)x_2$, where $\vartheta \in \mathbb{R}$, form the *line* passing through x_1 and x_2. The parameter value $\vartheta = 0$ corresponds to $y = x_2$ and $\vartheta = 1$ corresponds to $y = x_1$. Values of the parameter ϑ between 0 and 1 correspond to the *line segment* between x_1 and x_2.

Expressing y in the form $y = x_2 + \vartheta(x_1 - x_2)$ gives another interpretation: y is the sum of the base point x_2 (corresponding to $\vartheta = 0$) and the direction $x_1 - x_2$ (which points from x_2 to x_1) scaled by the parameter ϑ. Thus, ϑ gives the fraction of the way from x_2 to x_1 where y lies. As ϑ increases from 0 to 1, the point y moves from x_2 to x_1; for $\vartheta > 1$, the point y lies on the line beyond x_1. This is illustrated in Fig. B.1.

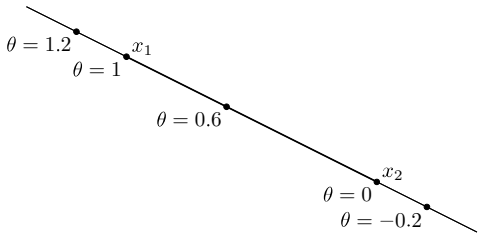


Figure B.1: The line passing through x_1 and x_2 is described parametrically by $\vartheta x_1 + (1-\vartheta)x_2$, where ϑ varies over $\mathbb{R}$. The line segment between x_1 and x_2, which corresponds to ϑ between 0 and 1, is shown darker (from [8])

A set $C \subseteq \mathbb{R}^n$ is *affine* if the line through any two distinct points in C lies in C, i.e., if for any $x_1,\ x_2 \in C$ and $\vartheta \in \mathbb{R}$, we have $\vartheta x_1 + (1-\vartheta)x_2 \in C$. C contains the linear combination of any two points in C, provided the coefficients in the linear combination sum to one.

The idea can be generalized to more than two points. We refer to a point of the form $\vartheta_1 x_1 + \cdots + \vartheta_k x_k$, where $\vartheta_1 + \cdots + \vartheta_k = 1$, as an *affine combination* of the points $x_1, \ldots, x_k$. Since an affine set contains every affine combination of two points in it, by induction it can be shown that an affine set contains every affine combination of its points: if C is an affine set, $x_1, \ldots, x_k \in C$, and $\vartheta_1 + \cdots + \vartheta_k = 1$, then the point $\vartheta_1 x_1 + \cdots + \vartheta_k x_k$ also belongs to C.

If C is an affine set and $x_0 \in C$, then the set $V = C - x_0 = \{x - x_0 | x \in C\}$ is a subspace (it is closed under sums and scalar multiplication). To see this, suppose $v_1, v_2 \in V$ and $\alpha, \beta \in \mathbb{R}$. Then we have $v_1 + x_0 \in C$ and $v_2 + x_0 \in C$, and so $\alpha v_1 + \beta v_2 + x_0 = \alpha(v_1 + x_0) + \beta(v_2 + x_0) + (1 - \alpha - \beta)x_0 \in C$, since C is affine, and $\alpha + \beta + (1 - \alpha - \beta) = 1$. We conclude that $\alpha v_1 + \beta v_2 \in V$, since $\alpha v_1 + \beta v_2 + x_0 \in C$. Thus, the affine set C can be expressed as $C = V + x_0 = \{v + x_0 | v \in V\}$, a subspace plus an offset. The subspace V associated with the affine set C does not depend on the choice of x_0, so x_0 can be chosen as any point in C. We define the dimension of an affine set C as the dimension of the subspace $V = C - x_0$, where x_0 is any element of C.

Example 1. *The solution set of a system of linear equations, $C = \{x | Ax = b\}$, where $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$, is an affine set. Conversely, every affine set can be expressed as the solution set of a system of linear equations.*

A set C is *convex* if the line segment between any two points in C lies in C: for any $x_1, x_2 \in C$ and any real ϑ with $0 \leq \vartheta \leq 1$, we have $\vartheta x_1 + (1-\vartheta)x_2 \in C$. Every affine set is also convex.

We call a point of the form $\vartheta_1 x_1 + \cdots + \vartheta_k x_k$, where $\vartheta_1 + \cdots + \vartheta_k = 1$ and $\vartheta_i \geq 0,\ i = 1, \ldots, k$, a *convex combination* of the points $x_1, \ldots, x_k$. As with affine sets, it can be shown that a set is convex if and only if it contains every convex combination of its points.

A set C is called a *cone*, or *nonnegative homogeneous*, if for every $x \in C$ and $\vartheta \geq 0$ we have $\vartheta x \in C$. A set C is a *convex cone* if it is convex and a cone: for any $x_1, x_2 \in C$ and $\vartheta_1, \vartheta_2 \geq 0$, we have $\vartheta_1 x_1 + \vartheta_2 x_2 \in C$. Points of this form can be described geometrically as

forming the two-dimensional pie slice with apex 0 and edges passing through x_1 and x_2. A point of the form $\vartheta_1 x_1 + \cdots + \vartheta_k x_k$ with $\vartheta_1, \ldots, \vartheta_k \geq 0$ is called a *conic combination* (or a *nonnegative linear combination*) of $x_1, \ldots, x_k$. If x_i are in a convex cone C, then every conic combination of x_i is in C. Conversely, a set C is a convex cone if and only if it contains all conic combinations of its elements.

The idea of affine, convex, conic combination can be generalized to infinite sums and integrals.

Some Examples

- The empty set $\emptyset$, any single point (singleton) $\{x_0\}$, and the whole space $\mathbb{R}^n$ are affine (hence, convex) subsets of $\mathbb{R}^n$.
- Any line is affine. If it passes through zero, it is a subspace, hence also a convex cone.
- A line segment is convex, but not affine (unless it reduces to a point).
- A *ray*, which has the form $\{x_0 + \vartheta v | \vartheta \geq 0\}$, where $v \neq 0$, is convex, but not affine. It is a convex cone if its base x_0 is 0.
- Any subspace is affine, and a convex cone (hence convex).
- A *hyperplane* is a set of the form $\{x | a^T x = b\}$, where $a \in \mathbb{R}^n$, $a \neq 0$, and $b \in \mathbb{R}$. Analytically it is the solution set of a nontrivial linear equation among the components of x (and hence an affine set). Geometrically, the hyperplane $\{x | a^T x = b\}$ can be interpreted as the set of points with a constant inner product to a given vector a (normal vector); the constant $b \in \mathbb{R}$ determines the offset of the hyperplane from the origin. The hyperplane can be expressed in the form $\{x | a^T(x - x_0) = 0\}$, where x_0 is any point in the hyperplane (any point that satisfies $a^T x_0 = b$). This representation can in turn be expressed as $\{x | a^T(x - x_0) = 0\} = x_0 + a^\perp$, where $a^\perp$ denotes the orthogonal complement of a, the set of all vectors orthogonal to it: $a^\perp = \{v | a^T v = 0\}$. Thus the hyperplane consists of an offset x_0, plus all vectors orthogonal to the normal vector a.
- A hyperplane divides $\mathbb{R}^n$ into two *halfspaces*. A *closed halfspace* is a set of the form $\{x | a^T x \leq b\}$, where $a \neq 0$: it is the solution set of a nontrivial linear inequality. Halfspaces are convex, but not affine. The halfspace can also be expressed as $\{x | a^T(x - x_0) \leq 0\}$, where x_0 is any point on the associated hyperplane (satisfies $a^T x_0 = b$). Hence a simple geometric interpretation: the halfspace consists of x_0 plus any vector that makes an obtuse angle with the outward normal vector a. The boundary of the halfspace is the hyperplane $\{x | a^T x = b\}$. The set $\{x | a^T x < b\}$, which is the interior of the halfspace $\{x | a^T x \leq b\}$, is called an *open halfspace*.
- A *(Euclidean) ball* in $\mathbb{R}^n$ has the form $B(x_c, r) = \{x | \, \|x - x_c\|_2 \leq r\} = \{x | (x - x_c)^T (x - xc) \leq r^2\}$, where $r > 0$, and $\|\cdot\|_2$ denotes the Euclidean norm, $\|u\|_2 = (u^T u)^{1/2}$. The vector x_c is the center of the ball and the scalar r is its radius; $B(x_c, r)$ consists of all points within a distance r of the center x_c. Another common representation for the Euclidean ball is $B(x_c, r) = \{x_c + ru | \, \|u\|_2 \leq 1\}$. A Euclidean ball is a convex set.
- Suppose $\|\cdot\|$ is any norm on $\mathbb{R}^n$; a norm ball of radius r and center x_c, given by $\{x | \, \|x - x_c\| \leq r\}$, is convex.
- A *polyhedron*, the intersection of a finite number of halfspaces and hyperplanes, is a convex set. Affine sets (subspaces, hyperplanes, lines, ...), rays, line segments, and halfspaces are all polyhedra.

Besides, a set C is convex if it is obtained from simple convex sets (hyperplanes, halfspaces, norm balls,...) by *operations that preserve convexity*, such as, for example, intersection and affine function (if f is an affine function, the image of a convex set under f is convex and the inverse image $f^{-1}(C)$ of a convex set under f is convex).

Convex Functions

A function $f : \mathbb{R}^n \to \mathbb{R}$ is *convex* if $dom(f)$ is a convex set and if for all $x, y \in dom(f)$, and ϑ with $0 \leq \vartheta \leq 1$, we have

$$f(\vartheta x + (1 - \vartheta)y) \leq \vartheta f(x) + (1 - \vartheta)f(y). \tag{B.3}$$

Geometrically, this inequality means that the line segment between $(x, f(x))$ and $(y, f(y))$, which is the chord from x to y, lies above the graph of f (Fig. B.2).

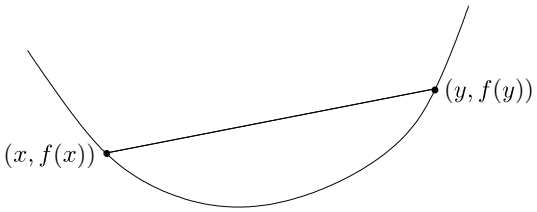


Figure B.2: Graph of a convex function: the chord between any two points on the graph lies above the graph (from [8])

A function f is strictly convex if strict inequality holds in (B.3) whenever $x \neq y$ and $0 < \vartheta < 1$. f is concave if $-f$ is convex, and strictly concave if $-f$ is strictly convex. For an affine function we always have equality in (B.3), so all affine functions are both convex and concave. Conversely, any function that is convex and concave is affine.

A function is convex if and only if it is convex when restricted to any line that intersects its domain: f is convex if and only if for all $x \in dom(f)$ and all v, the function $g(t) = f(x+tv)$ is convex (on its domain, $\{t|x + tv \in dom(f)\}$). This property is very useful, since it allows us to check whether a function is convex by restricting it to a line. The analysis of convex functions is a well developed field: one simple result, for example, is that a convex function is continuous on the relative interior of its domain; it can have discontinuities only on its relative boundary.

As for convex sets, there are operations that *preserve the convexity* of a function and allow us to construct new convex function from simple ones: e.g., nonnegative weighted sum, composition with affine function, pointwise maximum and supremum, scalar/vector composition, minimization.

Some conditions allow us to recognise a convex function.

First Order Condition

Suppose f is differentiable: its gradient

$$\nabla f = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_n} \right)$$

exists at each point in $dom(f)$, which is open. Then f is convex if and only if $dom(f)$ is convex and $f(y) \geq f(x) + \nabla f(x)^T(y - x)$ holds for all $x, y \in dom(f)$.

The affine function of y given by $f(x) + \nabla f(x)^T(y - x)$ is the first–order Taylor approximation of f near x. The inequality states that for a convex function, the first–order Taylor approximation is in fact a global underestimator of the function. Conversely, if the first–order Taylor approximation of a function is always a global underestimator of the function, then the function is convex. From local information about a convex function (its value and derivative at a point) we can derive global information (a global underestimator of it). This is perhaps the most important property of convex functions, and explains some of the remarkable properties of convex functions and convex optimization problems. As one simple example, the inequality shows that, if $\nabla f(x) = 0$, then for all $y \in dom(f)$, $f(y) \geq f(x)$: x is a

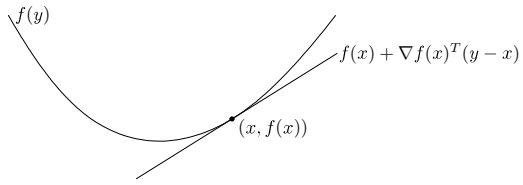


Figure B.3: If f is convex and differentiable, then $f(x) + \nabla f(x)^T(y - x) \leq f(y)$ for all $x, y \in dom(f)$ (from [8])

global minimizer of the function f. Strict convexity can also be characterized by a first–order condition: f is strictly convex if and only if $dom(f)$ is convex and for $x, y \in dom(f)$, $x \neq y$, we have $f(y) > f(x)+\nabla f(x)^T(y-x)$. For concave functions we have the corresponding characterization: f is concave if and only if $dom(f)$ is convex and $f(y) \leq f(x) + \nabla f(x)^T(y - x)$ for all $x, y \in dom(f)$.

Second Order Condition

We now assume that f is twice differentiable: its Hessian matrix or second derivative $\nabla^2 f$ exists at each point in $dom(f)$, which is open. Then f is convex if and only if $dom(f)$ is convex and its Hessian is positive semidefinite: for all $x \in dom(f)$, $\nabla^2 f(x) \geq 0$. For a function on $\mathbb{R}$, this reduces to the simple condition $f(x)'' \geq 0$ (and $dom(f)$ convex, i.e. an interval), which means that the derivative is nondecreasing. The condition $\nabla^2 f(x) \geq 0$ can be interpreted geometrically as the requirement that the graph of the function have positive (upward) curvature at x. Similarly, f is concave if and only if $dom(f)$ is convex and $\nabla^2 f(x) \leq 0$ for all $x \in dom(f)$. Strict convexity can be partially characterized by second–order conditions. If $\nabla^2 f(x) < 0$ for all $x \in dom(f)$, then f is strictly convex. The converse, however, is not true: for example, the function $f : \mathbb{R} \to \mathbb{R}$ given by $f(x) = x^4$ is strictly convex but has zero second derivative at $x = 0$.

Optimization Problems

We use the notation

$$\begin{array}{l} \text{minimize } f_0(x) \\ \text{subject to: } f_i(x) \leq 0,\ i = 1, \ldots, m \\ h_i(x) = 0,\ i = 1, \ldots, p \end{array} \tag{B.4}$$

to describe the problem of finding an x that minimizes $f_0(x)$ among all x that satisfy the conditions $f_i(x) \leq 0$, $i = 1, \ldots, m$ and $h_i(x) = 0$, $i = 1, \ldots, p$. $x \in \mathbb{R}^n$ is the *optimization variable* and the function $f_0 : \mathbb{R}^n \to \mathbb{R}$ the *objective function* or *cost function*. The inequalities $f_i(x) \leq 0$ are called *inequality constraints*, and the corresponding functions $f_i : \mathbb{R}^n \to \mathbb{R}$ *inequality constraint functions*. The equations $h_i(x) = 0$ are called the *equality constraints*, and the functions $h_i : \mathbb{R}^n \to \mathbb{R}$ *equality constraint functions*.

Remark 5. *Every equality constraint $h(x) = 0$ can be equivalently replaced by a pair of inequality constraints $h(x) \leq 0$ and $-h(x) \leq 0$. Therefore, for theoretical purposes, equality constraints are redundant; however, it can be beneficial to treat them specially in practice.*

If there are no explicit constraints ($m = p = 0$) the problem (B.4) is *unconstrained*. Yet every problem has an implicit constraint: the set of points for which the objective and all constraint functions are defined,

$$\mathcal{D} = \bigcap_{i=0}^{m} dom(f_i) \cap \bigcap_{i=1}^{p} dom(h_i),$$

is called the *domain* of the optimization problem (B.4). A point $x \in \mathcal{D}$ is feasible if it satisfies the constraints $f_i(x) \leq 0,\ i = 1, \ldots, m$ and $h_i(x) = 0,\ i = 1, \ldots, p$. The problem (B.4) is said to be *feasible* if there exists at least one feasible point, and *unfeasible* otherwise. The set of all feasible points is called the *feasible set* or the *constraint set*.

The optimal value p^* of the problem (B.4) is defined as $p^* = \inf\{f_0(x) | f_i(x) \leq 0,\ i = 1, \ldots, m,\ h_i(x) = 0,\ i = 1, \ldots, p\}$. We allow p^* to take on the extended values $\pm\infty$. If the problem is unfeasible, we have by convention $p^* = +\infty$. If there are feasible points x_k with $f_0(x_k) \to -\infty$ as $k \to \infty$, then $p^* = -\infty$, and the problem (B.4) is unbounded below.

Optimal and Locally Optimal Points

x^* is an *optimal point*, or solves the problem (B.4), if x^* is feasible and $f_0(x^*) = p^*$. The set of all optimal points is the *optimal set*, denoted $\mathcal{X}_{opt} = \{x | f_i(x) \leq 0,\ i = 1, \ldots, m,\ h_i(x) = 0,\ i = 1, \ldots, p,\ f_0(x) = p^*\}$. If there exists an optimal point for the problem (B.4), the optimal value is *attained* or *achieved*, and the problem is *solvable*. If $\mathcal{X}_{opt}$ is empty, the optimal value is not attained or not achieved. This always occurs when the problem is unbounded below. A feasible point x with $f_0(x) \leq p^* + \varepsilon$ (where $\varepsilon > 0$) is called *ε–suboptimal*, and the set of all ε–suboptimal points is called the ε–suboptimal set for the problem (B.4). A feasible point x is *locally optimal* if there is an $R > 0$ such that $f_0(x) = \inf\{f_0(z) | f_i(z) \leq 0,\ i = 1, \ldots, m,\ h_i(z) = 0,\ i = 1, \ldots, p,\ \|z - x\|_2 \leq R\}$, or, in other words, x solves the optimization problem

$$\begin{array}{l} \text{minimize } f_0(z) \\ \text{subject to: } f_i(z) \leq 0,\ i = 1, \ldots, m \\ h_i(z) = 0,\ i = 1, \ldots, p \\ \|z - x\|_2 \leq R \end{array}$$

with variable z. This means x minimizes f_0 over nearby points in the feasible set. The term 'globally optimal' is sometimes used for 'optimal' to distinguish between 'locally optimal' and 'optimal'.

A constraint is redundant if deleting it does not change the feasible set.

Example 2. *Here are a few simple unconstrained optimization problems with variable $x \in \mathbb{R}$, and $dom(f_0) = \mathbb{R}_{++}$:*

- *$f_0(x) = 1/x$: $p^* = 0$, but the optimal value is not achieved.*
- *$f_0(x) = -\log x$: $p^* = -\infty$, so this problem is unbounded below.*
- *$f_0(x) = x \log x$: $p^* = -1/e$, achieved at the (unique) optimal point $x^* = 1/e$.*

Feasibility Problems

If the objective function is identically zero, the optimal value is either zero (if the feasible set is nonempty) or ∞ (if the feasible set is empty). This is a *feasibility problem*, which can be written as

$$\begin{array}{l} \text{find } x \\ \text{subject to: } f_i(x) \leq 0,\ i = 1, \ldots, m \\ h_i(x) = 0,\ i = 1, \ldots, p \end{array}$$

The feasibility problem is thus to determine whether the constraints are consistent, and if so, find a point that satisfies them.

Standard Form

We refer to (B.4) as an optimization problem in standard form: the righthand side of the inequality and equality constraints are zero. This can always be arranged by subtracting any nonzero righthand side: we represent the equality constraint $g_i(x) = \tilde{g}_i(x)$, for example, as $h_i(x) = 0$, where $h_i(x) = g_i(x) - \tilde{g}_i(x)$. In a similar way we express inequalities of the form $f_i(x) \geq 0$ as $-f_i(x) \leq 0$.

Maximization Problems

We concentrate on the minimization problem by convention: the maximization problem

$$\begin{array}{l} \text{maximize } f_0(x) \\ \text{subject to: } f_i(x) \leq 0,\ i = 1, \ldots, m \\ h_i(x) = 0,\ i = 1, \ldots, p \end{array} \tag{B.5}$$

can be solved by minimizing the function $-f_0$ subject to the constraints. When the maximization problem is considered, the objective is sometimes called the *utility* or *satisfaction level* instead of the cost.

This shows that convex optimization has applications beyond minimizing convex functions: for instance, it is useful also for maximizing concave functions, since the problem of maximizing a concave function can be re–formulated equivalently as a problem of minimizing a convex function.

Equivalent Problems

Two problems can be considered *equivalent* if from a solution of one, a solution of the other is readily found, and vice versa. So it is useful to convert a problem to its equivalent standard form, and then solve it.

There are some general transformations that yield equivalent problems, such as change of variables, transformation of objective and constraint functions, eliminating/introducing equality constraints, introducing slack variables, optimizing over some variables.

The following problems (all treated in detail in [8]) are, or can be transformed into, convex optimization problems:

- least squares problems
- linear programming (LP)
- linear–fractional programming
- quadratic programming (QP)
- geometric programming (GP)
- second order cone programming (SOC)
- semidefinite programming (SDP)
- quadratically constrained quadratic programming (QCQP)
- entropy maximization

Convex Optimization Problem

A *convex optimization problem* can be written in the form

$$\begin{array}{l} \text{minimize } f_0(x) \\ \text{subject to: } f_i(x) \leq 0,\ i = 1, \ldots, m \\ a_i^T x = b_i,\ i = 1, \ldots, p \end{array}$$

or equivalently

$$\begin{array}{l} \text{minimize } f_0(x) \\ \text{subject to: } f_i(x) \leq 0,\ i = 1, \ldots, m \\ Ax = b \end{array}$$

where $f_0, f_1, \ldots, f_m$ are convex functions and equality constraints are affine.

Remark 6. *An important property is that the feasible set of a convex optimization problem is convex, since it is the intersection of the domain of the problem* $\mathcal{D} = \bigcap_{i=0}^{m} dom(f_i)$*, which is a convex set, with* m *(convex) sublevel sets* $\{x|f_i(x) \leq 0\}$ *and* p *hyperplanes* $\{x|a_i^T x = b_i\}$*. We can assume without loss of generality that* $a_i \neq 0$*: if* $a_i = 0$ *and* $b_i = 0$ *for some* i*, then the* i^{th} *equality constraint can be deleted; if* $a_i = 0$ *and* $b_i \neq 0$*, the* i^{th} *equality constraint is inconsistent, and the problem is unfeasible.*

Thus, in a convex optimization problem, we minimize a convex objective function over a convex set.

Another fundamental property of convex optimization problems is that any locally optimal point is also (globally) optimal.

Optimality criteria can be found for differentiable f_0, based upon ∇f_0.

Such results are described in [8], along with a more exhaustive treatment of the subject (involving the separating hyperplane theorem, the supporting hyperplane theorem, Farkas' lemma, duality theory, applications, methods and algorithms of resolution).

Software

The solution of convex optimization problems is found numerically and many software packages have been developed for this purpose, such as `CVX`. `CVX` uses the convex optimization solvers `SeDuMi` and `SDPT3`, all compatible with MATLAB environment (see [7]). They solve convex optimization problems via interior–point methods (developed by Nemirovski and Nesterov, see [8]).