ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

# CE1: NORMS OF SYSTEMS AND MODEL UNCERTAINTY

ADVANCED CONTROL SYSTEMS ME-524

PROF. ALIREZA KARIMI

---

GROUP B

Angelo Giovine 368440

Baptiste Bühler 326168

18th March 2024

# Contents

# CHAPTER 1

# NORMS OF SISO SYSTEMS

## 1.1

Our aim in this chapter is to compute the 2-Norm an the $\infty$-Norm using different techniques showing that the results is not changing.

The transfer function we are going to study is:

$$G(s) = \frac{s-1}{s^2+2s+10}$$

hence we wrote it in MATLAB:

**LISTING 1.1**
SISO System

```
G = tf([1 -1],[1 2 10]); % Transfer function
```

### 1.1.1 2-NORM

The 2-Norm of a transfer function, also known as the Euclidean norm measures the energy (or power) of the system response. For a transfer function $G(s)$, the 2-Norm is defined as the square root of the integral of the squared magnitude of the frequency response over all frequencies. Mathematically, it is represented as:

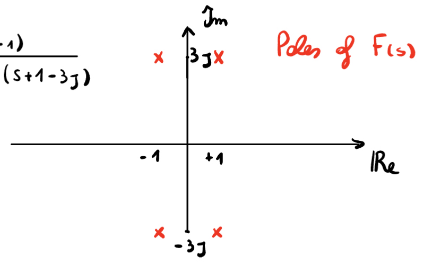
$$\|G\|_2 = \sqrt{\frac{1}{2\pi}\int_{-\infty}^{\infty} |G(j\omega)|^2 d\omega} \tag{1.1}$$

where $j$ is the imaginary unit and $\omega$ is the frequency in radians per second.

#### THE RESIDUE THEOREM

The solution is shown in Figure 1.1

2-Norm using residue theorem: $G(s) = \frac{s-1}{s^2+2s+10} = \frac{(s-1)}{(s+1+3j)(s+1-3j)}$




$F(s) = G(s) \cdot G(-s) = \frac{(s-1)(-s-1)}{(s+1+3j)(s+1-3j)(-s+1+3j)(-s+1-3j)}$

there are 2 poles in the LHP : $p_1 = -1+3j$ , $p_2 = -1-3j$

$\|G\|_2^2 = \sum_{k=1}^{2} Res[F(s), p_k] = Res[F(s), p_1] + Res[F(s), p_2]$

$Res[F(s), p_1] = \lim_{s \to (-1+3j)} \cancel{(s+1-3j)} \frac{(s-1)(-s-1)}{(s+1+3j)\cancel{(s+1-3j)}(-s+1+3j)(-s+1-3j)} = \frac{(3j-2)(-3j)}{(6j)\cdot(2)\cdot(2-6j)} = \frac{9+6j}{72+24j}$

$Res[F(s), p_2] = \lim_{s \to (-1-3j)} \cancel{(s+1+3j)} \frac{(s-1)(-s-1)}{\cancel{(s+1+3j)}(s+1-3j)(-s+1+3j)(-s+1-3j)} = \frac{(-2-3j)(3j)}{(-6j)\cdot(2+6j)\cdot(2)} = \frac{9-6j}{72-24j}$

$\|G\|_2 = \sqrt{\frac{9+6j}{72+24j} + \frac{9-6j}{72-24j}} = \frac{\sqrt{110}}{20} \simeq 0.5244$

**FIGURE 1.1**
Solution using the residue theorem

### THE FREQUENCY RESPONSE OF G

Given the symmetric nature of $|G(j\omega)|^2$, the 2-Norm can be efficiently computed through the equation:

$$\|G\|_2 = \sqrt{\left(\frac{1}{\pi}\int_0^\infty |G(j\omega)|^2 d\omega\right)} \tag{1.2}$$

Through the analysis of the Bode plot for the given system, an optimal frequency range for the integral approximation can be determined. While the system's dynamics are predominantly observed within the $10^{-2}$ to $10^{2}$ range, the integration bounds have been extended from $10^{-4}$ to $10^{4}$ to refine the 2-Norm estimation.

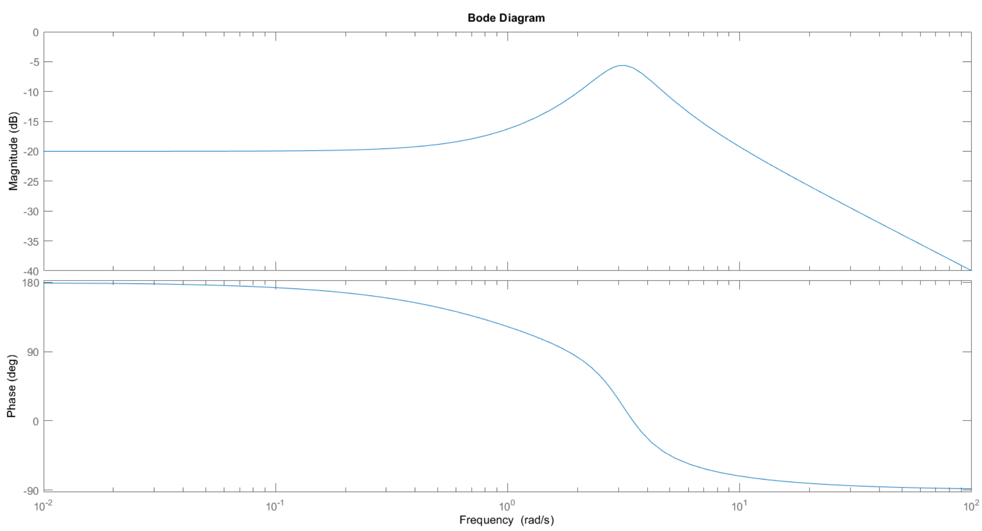



**FIGURE 1.2**
Bode plot of G(s)

The relative MATLAB code is : 1.2

**LISTING 1.2**
Frequency response

```
% Frequency response
omega = 10^(-4):0.01:10^(4); % Range of frequencies
Gjw = freqresp(G, omega);
dw = omega(2)-omega(1);
norm2_freq = sqrt((1/pi)*(sum((abs(Gjw).^2).*dw)));
```

### THE IMPULSE RESPONSE OF G

In this section, our objective is to calculate the 2-norm of $G$ through its impulse response representation. For a stable SISO system characterized by the transfer function $G(s)$, the 2-Norm can be articulated in terms of its impulse response $g(t)$, as shown below thanks to the Parseval's theorem:

$$\|G\|_2 = \sqrt{\int_{-\infty}^{\infty} |g(t)|^2 \, dt} \tag{1.3}$$

The relative MATLAB code is : 1.3

**LISTING 1.3**
Impulse response

```
% Impulse response
syms s
Numerator = poly2sym(G.Numerator{1,1},s);
Denominator = poly2sym(G.Denominator{1,1},s);
Gsym = Numerator/Denominator;
g = matlabFunction(ilaplace(Gsym)); % Inverse Laplace transform
norm2_impulse = sqrt(integral(@(t) (abs(g(t)).^2),0,Inf));
```

### THE STATE-SPACE METHOD

Our aim is to derive the 2-Norm of $G$ employing the state-space approach.
Utilizing the state-space technique, the 2-Norm of $G$ is discoverable via the equation:

$$\|G\|_2 = \sqrt{\text{trace}(CLC^T)} \tag{1.4}$$

wherein the matrix $L$ satisfies the condition:

$$AL + LA^T + BB^T = 0 \tag{1.5}$$

The relative MATLAB code is : 1.4

**LISTING 1.4**
State-space

```
% State-space
[A,B,C,D] = ssdata(G); % State-space matrices
L = are(A',zeros(2,2),B*B');
norm2_ss = sqrt(trace(C*L*C'));
```

#### VALIDATION WITH THE NORM FUNCTION

Using MATLAB's `norm` function, we can verify that our previous results are correct. 1.5

**LISTING 1.5**
true norm

```
true_norm2 = norm(G,2);
```

A comparison is shown in the following table:

| Method | Result |
|---|---|
| 2-Norm of $G$ using the residue theorem | 0.5244 |
| 2-Norm of $G$ using the frequency response | 0.5244 |
| 2-Norm of $G$ using the impulse response | 0.5244 |
| 2-Norm of $G$ using the state-space method | 0.5244 |
| Validation with `norm` function | 0.5244 |

### 1.1.2 $\infty$-NORM

The $\infty$-Norm of a transfer function, also known as the Maximum norm, measures the peak magnitude of the system's frequency response. It is defined as the maximum magnitude of the transfer function over all frequencies. Mathematically, it is represented as:

$$||G||_\infty = \sup_\omega |G(j\omega)| \tag{1.6}$$

#### THE FREQUENCY RESPONSE OF G

To compute the $\infty$-Norm we have chosen a fine grid of frequency points $\Omega$ looking at the Bode diagram in Figure 1.2, and solved this equation:

$$||G||_\infty \approx \max_{1\leq k\leq N} |G(j\omega_k)| \ , \ \Omega = \{\omega_1, ..., \omega_N\} \tag{1.7}$$

The relative MATLAB code is : 1.6

**LISTING 1.6**
Frequency response

```
% Frequency response
omega = 10^(-4):0.01:10^(4);  % Range of frequencies
Gjw = freqresp(G, omega);
normInf_freq = max(abs(Gjw));
```

#### THE BOUNDED REAL LEMMA

Here we want to compute the $\infty$-Norm of $G$ using the bounded real lemma.

**Lemma 1.1.1 (Bounded real lemma)** *Consider a strictly proper stable LTI system $G$ and $\gamma > 0$. Then $||G||_\infty < \gamma$ if and only if the Hamiltonian matrix $H$ has no eigenvalue on the imaginary axis:*

$$G(s) = \left[\begin{array}{c|c} A & B \\ \hline C & 0 \end{array}\right] H = \begin{bmatrix} A & \gamma^{-2}BB^T \\ -C^TC & -A^T \end{bmatrix}$$

Using this lemma we can implement an iterative bisection algorithm as follows: 1.7

**LISTING 1.7**
Bisection algorithm

```
% Bounded real lemma
gu = 1;
gl = 0.1;
eps = 1e-8;
while (gu-gl)/gl > eps
    g = (gu+gl)/2;
    H = [A g^(-2)*(B*B') ; -C'*C -A']; % Hamiltonian matrix
    if any(abs(real(eig(H))) < 1e-5) % Eigenvalue on the im axis
        gl = g;
    else % Eigenvalues not on the imaginary axis
        gu = g;
    end
end
normInf_lemma = (gu+gl)/2;
```

### VALIDATION WITH THE NORM FUNCTION

Using MATLAB's `norm` function, we can now verify that our previous results are correct.

**LISTING 1.8**
True 2-norm

```
true_normInf = norm(G,inf);
```

The outcomes for each technique are presented below:

| Method | Result |
|---|---|
| $\infty$-Norm of $G$ using the frequency response | 0.5246 |
| $\infty$-Norm of $G$ using the bounded real lemma | 0.5246 |
| Validation with `norm` function | 0.5246 |

# CHAPTER 2

# NORM OF A MIMO SYSTEM

## 2.1

In this chapter we are going to study the 2-Norm and the $\infty$-Norm of the following MIMO system:

$$\dot{x} = Ax + Bu$$

$$y = Cx + Du$$

with

$$A = \begin{bmatrix} 20 & -27 & 7 \\ 53 & -63 & 13 \\ -5 & 12 & -8 \end{bmatrix}, \quad B = \begin{bmatrix} 1 & -1 \\ -2 & -1 \\ -3 & 0 \end{bmatrix}, \quad C = \begin{bmatrix} 0 & 0 & -2 \\ 1 & -1 & -1 \end{bmatrix}, \quad D = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

So we upload the system in MATLAB:

**LISTING 2.1**
MIMO System

```
A = [20 -27 7 ; 53 -63 13 ; -5 12 -8];
B = [1 -1 ; -2 -1 ; -3 0];
C = [0 0 -2 ; 1 -1 -1];
D = zeros(2,2);
sys = ss(A,B,C,D);
G = tf(sys);
```

### 2.1.1 2-NORM

#### THE FREQUENCY RESPONSE METHOD

Denoting the MIMO system as $G(s)$ we can use the following equation, where $G^*$ denotes the complex conjugate transpose of $G$, to compute the 2-Norm:

$$||G||_2 = \sqrt{\left(\frac{1}{2\pi}\int_{-\infty}^{\infty} \text{trace}[G^*(j\omega)G(j\omega)]\right)} \tag{2.1}$$

Since trace$[G^*(j\omega)G(j\omega)]$ is symmetric, we note that instead of computing the integral in equation (2.1), we can use the equation:

$$||G||_2 = \sqrt{\left(\frac{1}{\pi}\int_0^\infty \text{trace}[G^*(j\omega)G(j\omega)]\right)} \tag{2.2}$$

Analogous to the approach employed for Single-Input Single-Output (SISO) systems, we analyze the Bode plots to choose the optimal frequency spectrum for the numerical integration.The core behaviors of the system are noticeable within the frequency range from $10^{-3}$ to $10^3$ Hz Figure 2.1. However, to improve the accuracy of the 2-Norm calculation, the integration boundaries have been expanded to encompass a wider frequency range, from $10^{-4}$ to $10^5$ Hz.

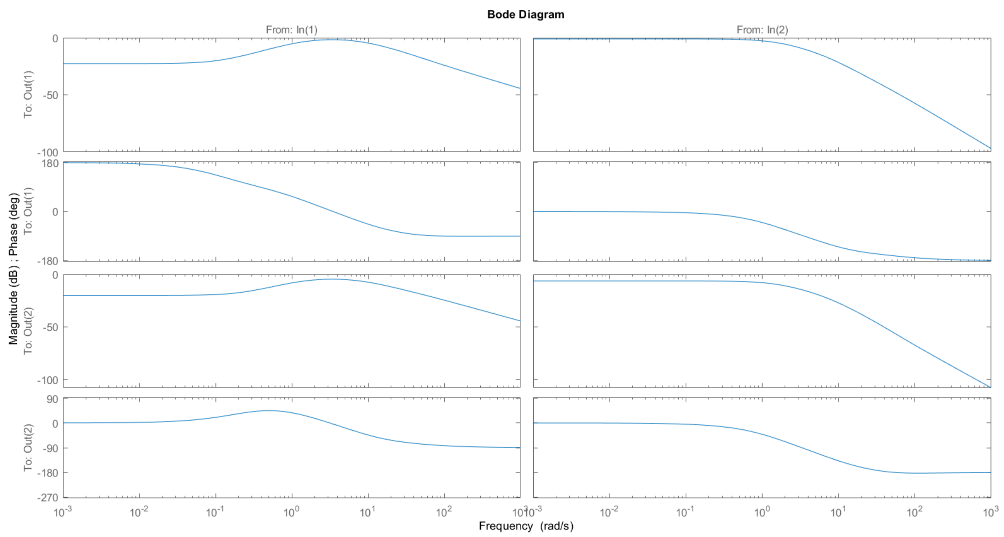



**FIGURE 2.1**
Bode diagram of the MIMO system

The relative MATLAB code is : 2.2

**LISTING 2.2**
Frequency response MIMO

```
% Frequency response
omega = logspace(-4,5,1000);  % Range of frequencies
Gjw = [];
for i = 1:size(omega,2)
    Gjw = [Gjw, trace(freqresp(conj(G)'*G, omega(i)))];
end
norm2_freq = abs(sqrt(1/pi*trapz(omega, Gjw)));
```

### THE STATE-SPACE METHOD

The considerations made for the SISO system 1.1.1 are the same for the MIMO one hence:

**LISTING 2.3**
State space MIMO

```
% State-space
```

```
L = are(A',zeros(3,3),B*B');
norm2_ss = sqrt(trace(C*L*C'));
```

### VALIDATION WITH THE NORM FUNCTION

Once again we validate the result through the `norm` function:

**LISTING 2.4**
True 2 Norm MIMO

```
% True value
true_norm2 = norm(G,2);
```

A summary of the results is shown below:

| Method | Result |
|---|---|
| 2-Norm of $G$ using the frequency response | 2.2818 |
| 2-Norm of $G$ using the state-space method | 2.2818 |
| Validation with `norm` function | 2.2818 |

## 2.1.2 $\infty$-NORM

### THE FREQUENCY RESPONSE METHOD

To compute the $\infty$-Norm of $G$, we use the following equation:

$$||G||_\infty = \sup_\omega ||G(j\omega)|| = \sup_\omega \bar{\sigma}[G(j\omega)] = \sup_\omega \sqrt{\lambda_{max}(G(j\omega)^* G(j\omega))} \tag{2.3}$$

Once again we chose the range of frequencies as stated in 2.1.1

The relative MATLAB code is : 2.5

**LISTING 2.5**
Frequency response MIMO

```
% Frequency response
omega = logspace(-4,5,1000);  % Range of frequencies
Gjw = freqresp(conj(G)'*G, omega);
eigen = [];
for i = 1:size(omega,2)
    eigen = [eigen, real(eig(freqresp(conj(G)'*G,omega(i))))];
end
normInf_freq = max(sqrt(max(eigen)));
```

### THE BOUNDED REAL LEMMA

The explanation provided in 1.1.2 does not depend on the assumption that the system is SISO. Therefore, both the lemma and the algorithm can be adapted for MIMO systems.The relative MATLAB code is : 2.6

**LISTING 2.6**
Bisection MIMO

```
% Bounded real lemma
gu = 3;
```

```
gl = 0.1;
eps = 1e-8;
while (gu-gl)/gl > eps
    g = (gu+gl)/2;
    H = [A g^(-2)*(B*B') ; -C'*C -A']; % Hamiltonian matrix
    if any(abs(real(eig(H))) < 1e-5) % Eigenvalue on the ima axis
        gl = g;
    else % Eigenvalues not on the imaginary axis
        gu = g;
    end
end
normInf_lemma = (gu+gl)/2;
```

### VALIDATION WITH THE NORM FUNCTION

Using MATLAB's `norm` function, we can now verify that our previous results are correct.

**LISTING 2.7**
True Inf norm MIMO

```
% True value
true_normInf = norm(G,inf);
```

A comparison is shown in the following table:

| **Method** | **Result** |
|---|---|
| $\infty$-Norm of $G_{\text{MIMO}}$ using the frequency response | 1.1081 |
| $\infty$-Norm of $G_{\text{MIMO}}$ using the bounded real lemma | 1.1081 |
| Validation with `norm` function | 1.1081 |

# Chapter 3

# Uncertainty modeling

## 3.1

In this chapter the goal is to derive the weighting filter for the following parametric uncertainty model:

$$G(s) = \frac{a}{s^2 + bs + c}$$

where $a = 11 \pm 1$, $b = 4 \pm 1$ and $c = 9 \pm 2$.

The weighting filter $W_2(s)$ is a filter used to assess the robustness of an uncertain model. Let's consider a transfer function $\tilde{G}(s)$ which can be decomposed into $\tilde{G}(s) = G(1 + \Delta(s)W_2(s))$ where $\Delta(s)$ is the error with $\|\Delta\|_\infty \leq 1$ and $G(s)$ is the nominal model of $\tilde{G}(s)$.

So we define the transfer function $\tilde{G}(s)$, using the MATLAB's `ureal` function, and its nominal value in MATLAB. Here we have arbitrary chosen $a = 11$, $b = 4$ and $c = 9$ for the nominal system. 3.1

**Listing 3.1**
Uncertainty model

```
% Uncertainty model
a = ureal('a',11,'PlusMinus',1);
b = ureal('b',4,'PlusMinus',1);
c = ureal('c',9,'PlusMinus',2);
G = tf(a,[1 b c]);
nominalG = G.NominalValue;
```

We can observe the step response, the Bode diagram and the Nyquist plot of $G(s)$ in Figure 3.1:

Then we must sample the transfer function $G(s)$ twice to obtain a number $n \in \{20,\ 200\}$ of models using the MATLAB's `usample` function. The relative MATLAB code is:

**Listing 3.2**
Multimodel uncertainty

```
% Multimodel uncertainty
usys20 = usample(G,20);
usys200 = usample(G,200);
```

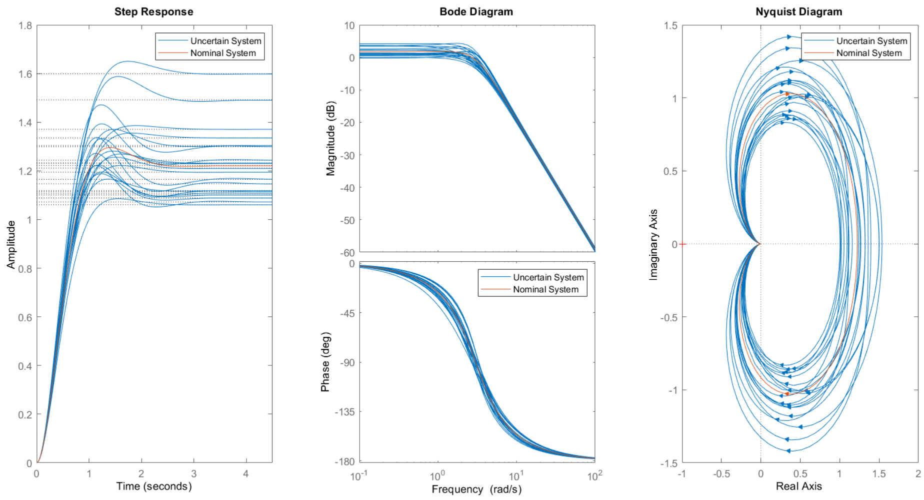



**FIGURE 3.1**
Step response, Bode diagram and Nyquist plot of the uncertain system

When we have a multimodel uncertainty $\mathcal{G} = \{G_1, G_2, \cdots, G_n\}$ it is possible to convert into a multiplicative uncertainty. We know the nominal model of the system $G$, thus:

$$\tilde{G} = G(1 + \Delta W_2) \Rightarrow \frac{G_i}{G} - 1 = \Delta W_2 \qquad \text{for } i = 1, \cdots, n$$

Since $\|\Delta\|_\infty \leq 1 \Rightarrow \left|\frac{G_i(j\omega)}{G(j\omega)} - 1\right| \leq |\Delta W_2(j\omega)| \qquad \text{for } i = 1, \cdots, n$

Then the aim is to compute $\bar{W}_2(j\omega)$ such that $\left|\bar{W}_2(j\omega)\right| = \max_i \left|\frac{G_i(j\omega)}{G(j\omega)} - 1\right| \ \forall\omega$

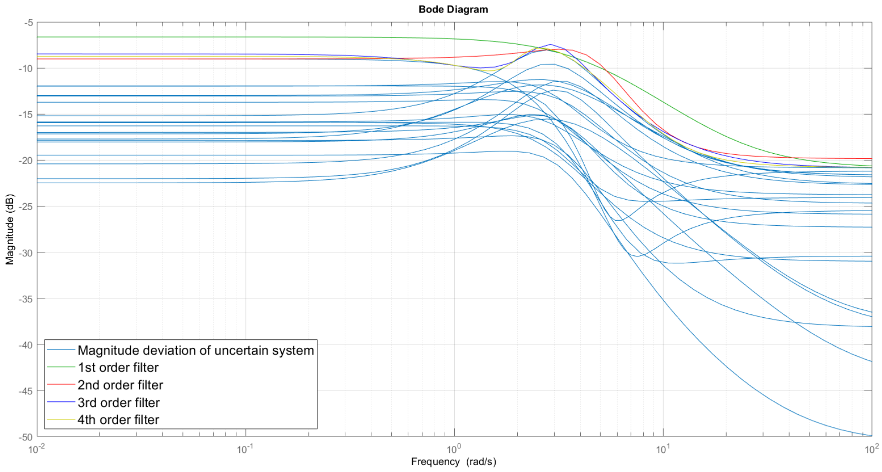
Using the MATLAB's `ucover` function, we can convert the multimodel uncertainty set to a multiplicative one. We assume that the nominal value of each multimodel uncertainty sets is equal to the nominal model defined above in 3.1Then it is possible to obtain the corresponding weighting filter $\bar{W}_2$ for each multimodel set. The relative MATLAB code is:

**LISTING 3.3**
Multiplicative uncertainty

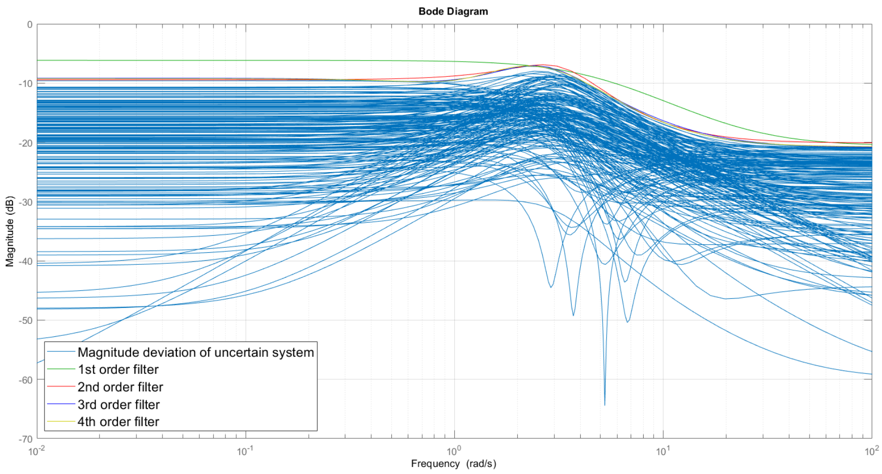
```
% Convert to multiplicative uncertainty
[~,Info_20] = ucover(usys20, nominalG, 1); % filter of order 1
[~,Info_200] = ucover(usys200, nominalG, 1);
[~,Info2_20] = ucover(usys20, nominalG, 2); % filter of order 2
[~,Info2_200] = ucover(usys200, nominalG, 2);
[~,Info3_20] = ucover(usys20, nominalG, 3); % filter of order 3
[~,Info3_200] = ucover(usys200, nominalG, 3);
[~,Info4_20] = ucover(usys20, nominalG, 4); % filter of order 4
[~,Info4_200] = ucover(usys200, nominalG, 4);
```

The results are given in the Figure 3.2 where the relative errors $\frac{G_i}{G} - 1$ for $i = 1, \cdots, n$ are plotted.

Of course the weighting filter depends on the chosen samples 3.2




(A) 20 Samples




(B) 200 Samples

**FIGURE 3.2**
Weighting filter for each multimodel uncertainty

Here we have reported the expression of the second order approximation, while for the plot we have tested different orders approximation:

$$\text{20 samples second order: } W_2(s) = \frac{0.09056s^2 + 0.9606s + 4.075}{s^2 + 3.948s + 11.12}$$

$$\text{200 samples second order: } W_2(s) = \frac{0.09279s^2 + 1.532s + 5.812}{s^2 + 4.392s + 16.09}$$

We can compare the the different filters in the Bode diagram shown in Figure 3.3 and Figure 3.4. The initial and final value are quite similar and clearly the weighting filter of the 200-samples multimodel has an higher magnitude because the more the samples the more representative is the model. Moreover is evident the effect of an higher order approximation that at the cost of a bigger computational cost provide a better upper bound for our systems.

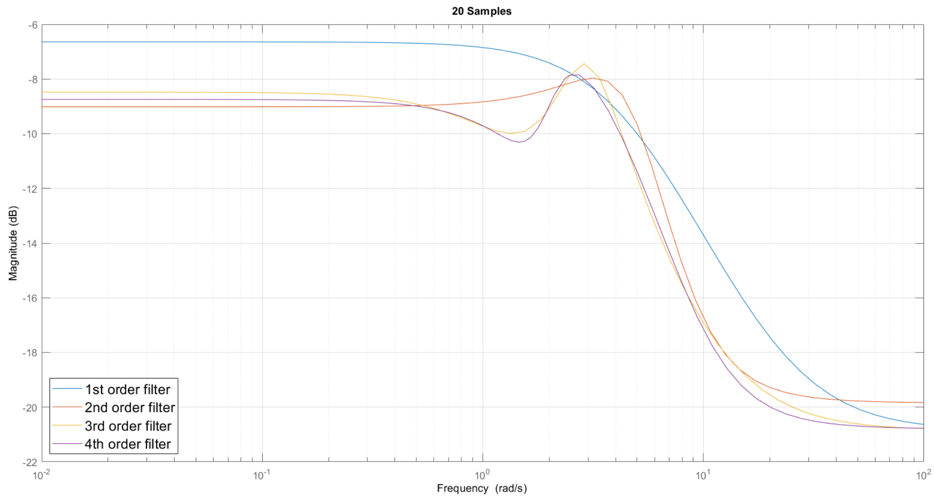



**FIGURE 3.3**
Bode diagram of weighting filters, 20 samples.

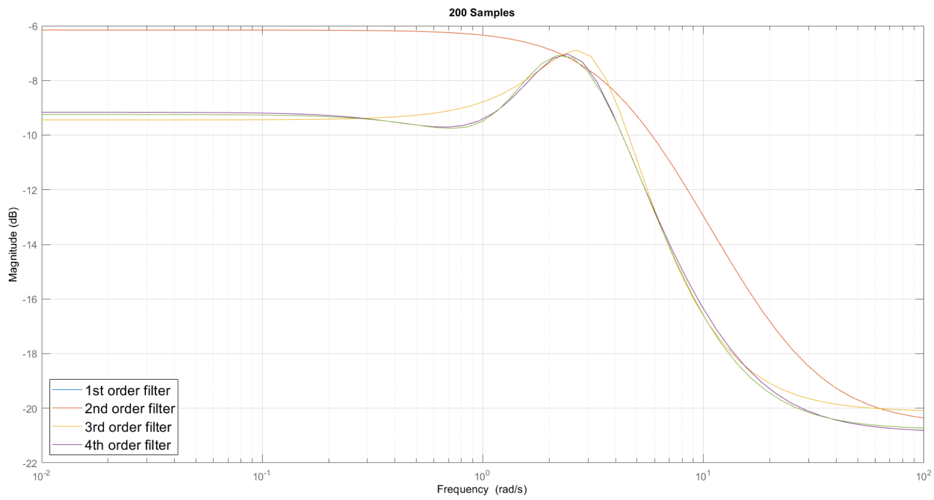



**FIGURE 3.4**
Bode diagram of weighting filters, 200 samples.

# Appendix A

# MATLAB's code

Here is the whole MATLAB's code used in this computer exercise.

**Listing A.1**
Computer exercise 1

```
clc, clear, close all

%% 1.1
fprintf('SISO\n')
G = tf([1 -1],[1 2 10]); % Transfer function

% 1.1.1
% Frequency response approximative
omega = 0:0.01:10000;  % Range of frequencies
Gjw = freqresp(G, omega);
dw = omega(2)-omega(1);
norm2_freq = sqrt((1/pi)*(sum((abs(Gjw).^2).*dw)));

% Impulse response
syms s
Numerator = poly2sym(G.Numerator{1,1},s);
Denominator = poly2sym(G.Denominator{1,1},s);
Gsym = Numerator/Denominator;
g = matlabFunction(ilaplace(Gsym)); % Inverse Laplace transform
norm2_impulse = sqrt(integral(@(t) (abs(g(t)).^2),0,Inf));

% State-space
[A,B,C,D] = ssdata(G); % State-space matrices
L = are(A',zeros(2,2),B*B');
norm2_ss = sqrt(trace(C*L*C'));

% True value
true_norm2 = norm(G,2);

fprintf(['H_2: \n' ...
    'Frequency response => %.4f\n' ...
```

```
    'Impulse response => %.4f\n' ...
    'State-space method => %.4f\n' ...
    'True value => %.4f\n\n'], norm2_freq, norm2_impulse, norm2_ss,
       true_norm2)

% 1.1.2
% Frequency response
omega = 0:0.001:100;  % Range of frequencies
Gjw = freqresp(G, omega);
normInf_freq = max(abs(Gjw));

% Bounded real lemma
gu = 1;
gl = 0.1;

eps = 1e-8;

while (gu-gl)/gl > eps
    g = (gu+gl)/2;

    H = [A g^(-2)*(B*B') ; -C'*C -A']; % Hamiltonian matrix

    if any(abs(real(eig(H))) < 1e-5) % Eigenvalue on the imaginary
       axis
        gl = g;
    else % Eigenvalues not on the imaginary axis
        gu = g;
    end
end

normInf_lemma = (gu+gl)/2;

% True value
true_normInf = norm(G,inf);

fprintf(['H_Inf: \n' ...
    'Frequency response => %.4f\n' ...
    'Impulse response => %.4f\n' ...
    'True value => %.4f\n\n'], normInf_freq, normInf_lemma,
       true_normInf)
%% 1.2
fprintf('MIMO\n')

A = [20 -27 7 ; 53 -63 13 ; -5 12 -8];
B = [1 -1 ; -2 -1 ; -3 0];
C = [0 0 -2 ; 1 -1 -1];
D = zeros(2,2);

sys = ss(A,B,C,D);
```

```

G = tf(sys);

% 1.2.1
% Frequency response
omega = logspace(-4,5,1000);  % Range of frequencies
Gjw = [];
for i = 1:size(omega,2)
    Gjw = [Gjw, trace(freqresp(conj(G)'*G, omega(i)))];
end
norm2_freq = abs(sqrt(1/pi*trapz(omega, Gjw)));

% State-space
L = are(A',zeros(3,3),B*B');
norm2_ss = sqrt(trace(C*L*C'));

% True value
true_norm2 = norm(G,2);

fprintf(['H_2: \n' ...
    'Frequency response => %.4f\n' ...
    'State-space method => %.4f\n' ...
    'True value => %.4f\n\n'], norm2_freq, norm2_ss, true_norm2)

% 1.2.2
% Frequency response
omega = logspace(-4,5,1000);  % Range of frequencies
Gjw = freqresp(conj(G)'*G, omega);
eigen = [];
for i = 1:size(omega,2)
    eigen = [eigen, real(eig(freqresp(conj(G)'*G,omega(i))))];
end
normInf_freq = max(sqrt(max(eigen)));

% Bounded real lemma
gu = 3;
gl = 0.1;

eps = 1e-8;

while (gu-gl)/gl > eps
    g = (gu+gl)/2;

    H = [A g^(-2)*(B*B') ; -C'*C -A']; % Hamiltonian matrix

    if any(abs(real(eig(H))) < 1e-5) % Eigenvalue on the imaginary
       axis
        gl = g;
    else % Eigenvalues not on the imaginary axis
```

```
        gu = g;
    end
end

normInf_lemma = (gu+gl)/2;

% True value
true_normInf = norm(G,Inf);

fprintf(['H_Inf: \n' ...
    'Frequency response => %.4f\n' ...
    'Impulse response => %.4f\n' ...
    'True value => %.4f\n\n'], normInf_freq, normInf_lemma,
       true_normInf)

%% 1.3
% Uncertain model
a = ureal('a',11,'PlusMinus',1);
b = ureal('b',4,'PlusMinus',1);
c = ureal('c',9,'PlusMinus',2);
G = tf(a,[1 b c]);

nominalG = G.NominalValue;

% Bode, Nyquist and step response
figure(1)
subplot(1,3,1)
step(G)
hold on
step(nominalG)
legend('Uncertain System','Nominal System')
subplot(1,3,2)
bode(G)
hold on
bode(nominalG)
legend('Uncertain System','Nominal System')
subplot(1,3,3)
nyquist(G)
hold on
nyquist(nominalG)
legend('Uncertain System','Nominal System')

% Multimodel uncertainty
usys20 = usample(G,20);
usys200 = usample(G,200);

% Convert to multiplicative uncertainty
[~,Info_20] = ucover(usys20, nominalG, 1); % filter of order 1
[~,Info_200] = ucover(usys200, nominalG, 1);
```

```
[~,Info2_20] = ucover(usys20, nominalG, 2); % filter of order 2
[~,Info2_200] = ucover(usys200, nominalG, 2);
[~,Info3_20] = ucover(usys20, nominalG, 3); % filter of order 3
[~,Info3_200] = ucover(usys200, nominalG, 3);
[~,Info4_20] = ucover(usys20, nominalG, 4); % filter of order 4
[~,Info4_200] = ucover(usys200, nominalG, 4);

figure(3); % Plot for 20 samples
hold on
bodemag(usys20/nominalG -1);
bodemag(Info_20.W1, '-g');% filter of order 1
bodemag(Info2_20.W1, '-r'); % filter of order 2
bodemag(Info3_20.W1, '-m'); % filter of order 3
bodemag(Info4_20.W1, '-y'); % filter of order 4
xlim([10^(-2) 10^2])
legend('Magnitude deviation of uncertain system', '1st order
    filter', '2nd order filter', '3rd order filter', '4th order
    filter', 'fontsize', 14, 'Location', 'SouthWest')
grid on

figure(4); % Plot for 200 samples
hold on
bodemag(usys200/nominalG -1);
bodemag(Info_200.W1, '-g'); % filter of order 1
bodemag(Info2_200.W1, '-r'); % filter of order 2
bodemag(Info3_200.W1, '-m'); % filter of order 3
bodemag(Info4_200.W1, '-y'); % filter of order 4
xlim([10^(-2) 10^2])
legend('Magnitude deviation of uncertain system', '1st order
    filter', '2nd order filter', '3rd order filter', '4th order
    filter', 'fontsize', 14, 'Location', 'SouthWest')
grid on

figure(5)
hold on
bodemag(ss(Info_20.W1.A,Info_20.W1.B,Info_20.W1.C,Info_20.W1.D))
bodemag(ss(Info2_20.W1.A,Info2_20.W1.B,Info2_20.W1.C,Info2_20.W1.D))
bodemag(ss(Info3_20.W1.A,Info3_20.W1.B,Info3_20.W1.C,Info3_20.W1.D))
bodemag(ss(Info4_20.W1.A,Info4_20.W1.B,Info4_20.W1.C,Info4_20.W1.D))
xlim([10^(-2) 10^2])
legend( '1st order filter', '2nd order filter', '3rd order filter',
    '4th order filter', 'fontsize', 14, 'Location', 'SouthWest')
title('20 Samples')
grid on

figure(6)
hold on
bodemag(ss(Info_200.W1.A,Info_200.W1.B,Info_200.W1.C,Info_200.W1.D))
bodemag(ss(Info2_200.W1.A,Info2_200.W1.B,Info2_200.W1.C,Info2_200.W1.D))
```

```
bodemag(ss(Info3_200.W1.A,Info3_200.W1.B,Info3_200.W1.C,Info3_200.W1.D))
bodemag(ss(Info4_200.W1.A,Info4_200.W1.B,Info4_200.W1.C,Info4_200.W1.D))
xlim([10^(-2) 10^2])
legend( '1st order filter', '2nd order filter', '3rd order filter',
   '4th order filter', 'fontsize', 14, 'Location', 'SouthWest')
title('200 Samples')
grid on
```