Advanced Control Systems
Computer Exercise 1

Gabelli Léo, Vignoli Lorenzo

403831, 405238

March 16, 2025

1 Norms of SISO systems

Consider the following second-order model, whose Bode diagram is shown in Figure 1:

$$G(s) = \frac{s - 10}{s^2 + 5s + 26.5}$$

Several methods are used to compute the $\mathcal{H}_2$ and $\mathcal{H}_\infty$ norms.

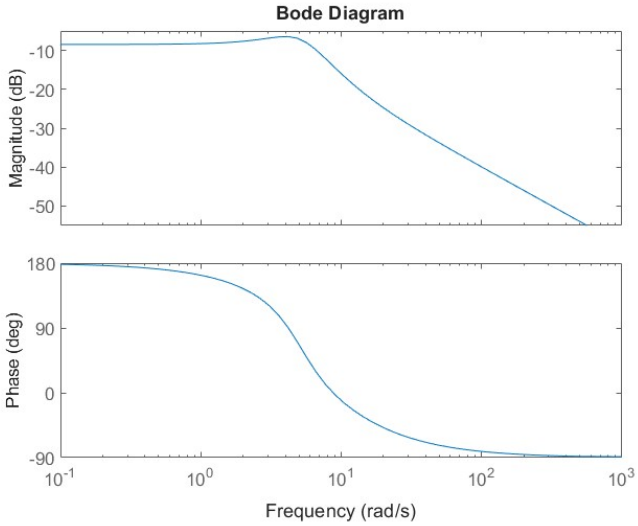

Figure 1: Bode diagram of G

1.1 2-Norm

Four different methods are employed:

1. *Residue theorem*: the system has two poles p_1 and p_2. Computing the limits, the corresponding residues are:

$$\text{Res}[G(-s)G(s), p_1] = \frac{(p_1 - 10)(-p_1 - 10)}{(p_1 - p_2)(-p_1 - p_2)(-2 \cdot p_1)}$$

$$\text{Res}[G(-s)G(s), p_2] = \frac{(p_2 - 10)(-p_2 - 10)}{(p_2 - p_1)(p_2 + p_1)(2 \cdot p_2)}$$

Consequently, the $\mathcal{H}_2$-norm is computed as:

$$\|G\|_2 = \sqrt{\sum_{k=1}^{2} \text{Res}[G(-s)G(s), p_k]}$$

2. *Frequency response of G*: the $\mathcal{H}_2$-norm can be approximated by numerical integration using the trapezoidal method in the code:

$$\|G\|_2 = \sqrt{\frac{1}{\pi} \sum_{k=0}^{N} |G(j\omega_k)|^2 \Delta\omega_k}$$

where N is chosen large enough to account for all meaningful contributions with respect to Figure 1, i.e., $|G(j\omega_k)|$ can be ignored $\forall k > N$.

3. *Impulse response of G*: thanks to Parseval's theorem, and given that the system is stable, the $\mathcal{H}_2$-norm can be computed using its impulse response $g(t)$:

$$\|G\|_2 = \sqrt{\sum_{k=0}^{N} |g(t_k)|^2 \Delta t_k}$$

where the impulse response is evaluated over an arbitrarily long time period (10 minutes).

4. *State-space method*: considering the state-space representation, equivalent to $G(s)$,

$$\dot{x} = Ax + Bu, \quad y = Cx$$

the algebraic Riccati equation is solved:

$$AL + LA^T + BB^T = 0$$

and the $\mathcal{H}_2$-norm is given by:

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

It is interesting to note that in MATLAB, the Riccati equation is solved using the function `are`, whose input matrices must be defined accordingly.

Table 1: Computed 2-Norms

Computation	2-Norm
Residue theorem	0.69091
Frequency response of G	0.69045
Impulse response of G	0.69098
State-space method	0.69091
MATLAB command `norm`	0.69091

As seen in Table 1, the results are consistent with those obtained using the MATLAB command `norm`. The least accurate results are those obtained with the frequency and impulse response methods; however, a finer discretization or a larger domain could easily reduce the discrepancy.

1.2 ∞-Norm

Two different methods are employed:

1. *Frequency response of G*: the $\mathcal{H}_\infty$-norm is computed as:

$$\|G\|_\infty = \max_{1 \le k \le N} |G(j\omega_k)|$$

2. *Bounded real lemma*: an iterative bisection algorithm is used, starting from upper and lower bounds γ_u and γ_l, and halving the interval by checking each time whether the Hamiltonian matrix

$$H = \begin{bmatrix} A & \gamma^{-2} BB^T \\ -C^T C & -A^T \end{bmatrix}$$

has eigenvalues on the imaginary axis (i.e., their real parts are below a very small threshold chosen for numerical reasons).

Table 2: Computed ∞-Norms

Computation	∞-Norm
Frequency response of G	0.47682
Bounded real lemma	0.47682
MATLAB command `norm`	0.47682

Table 2 shows that the MATLAB command `norm` consistently yields the same results.

1.3 MATLAB code

```
%% 1.1 Norm of SISO systems

clc
close all
clear all
num = [1 -10];
den = [1 5 26.5];
G = tf(num, den);   % second-order model
```

```
figure()
bode(G);

% 1.1.1 2-NORM
% 1. Residue theorem (sum of two residuals)
poles = roots(den);
p1 = poles(1);
p2 = poles(2);
Res1 = (p1-10)*(-p1-10)/((p1-p2)*(-p1-p2)*(-2*p1));
Res2 = (p2-10)*(-p2-10)/((p2-p1)*(p2+p1)*(p2*2));
Norm1 = sqrt(Res1+Res2);
disp(['2-norm = ', num2str(Norm1)]);

% 2. Frequency response of G (integral)
omega = linspace(0,5e2,1e5);    % relevant interval
G_jomega = (j*omega-10)./((j*omega).^2+5.*j*omega+26.5);
Norm2 = sqrt(1/pi*trapz(omega, abs(G_jomega).^2));
disp(['2-norm = ', num2str(Norm2)]);

% 3. Impulse response of G
t = linspace(0, 600, 1e5);  % time
g = impulse(G, t);
Norm3 = sqrt(trapz(t, abs(g).^2));
disp(['2-norm = ', num2str(Norm3)]);

% 4. State-space method
[A,B,C,D] = ssdata(G);
L = are(A', zeros(2), B*B');
Norm4 = sqrt(trace(C*L*C'));
disp(['2-norm = ', num2str(Norm4)]);

% 5. Validate results with MATLAB command norm()
disp(['2-norm = ', num2str(norm(G))]);

% 1.1.2. INFINITY-NORM
% 1. Frequency response of G
Norm5 = max(abs(G_jomega));
disp(['Infinity-norm = ', num2str(Norm5)]);

% 2. Bounded real lemma (iterative bisection algorithm)
gamma_u = 1;
gamma_l = 1e-5;     % boundaries
eps = 1e-8;
flag = true;
niter = 0;

while flag
    if (gamma_u-gamma_l)/gamma_l < eps
        flag = false;
        Norm6 = mean([gamma_u, gamma_l]);
    else
    gamma = mean([gamma_u, gamma_l]);
    H = [A, gamma^(-2)*B*B'; -C'*C, -A'];
    if any(abs(real(eig(H)))<1e-3)
        gamma_l = gamma;
    else
```

```
        gamma_u = gamma;
    end
    end
    niter = niter + 1;
end
disp(['Infinity-norm = ', num2str(Norm6)]);

% 3. Validate results with norm()
disp(['Infinity-norm = ', num2str(norm(G, inf))]);
```

2 Norms of MIMO systems

Consider the following state-space model of a MIMO system:

$$\dot{x} = Ax + Bu, \quad y = Cx$$

where

$$A = \begin{bmatrix} -8.0 & 2.5 & -2.5 \\ 7.8 & -3.2 & 13.8 \\ 0.1 & -0.5 & -5.7 \end{bmatrix}, \quad B = \begin{bmatrix} 1 & 0 \\ 1 & -1 \\ 0 & -2 \end{bmatrix}, \quad C = \begin{bmatrix} 0 & 0 & 1 \\ 1 & -1 & 0 \end{bmatrix}.$$

The corresponding transfer function is obtained using the MATLAB function `ss`, i.e.,

$$G(s) = C(sI - A)^{-1}B.$$

Several methods are then used to compute the $\mathcal{H}_2$ and $\mathcal{H}_\infty$ norms.

2.1 2-Norm

Two different methods are employed:

1. *Frequency response method*: the norm is obtained by numerical integration:

$$\|G\|_2 = \sqrt{\frac{1}{\pi}\sum_{k=0}^{N} \text{trace}\left(G^*(j\omega_k)G(j\omega_k)\right)\Delta\omega_k}$$

   where $G(j\omega_k)$ is derived from the aforementioned formula, and the trace is numerically integrated using the trapezoidal rule.

2. *State-space method*: analogously to the SISO case, the algebraic Riccati equation is solved to determine the $\mathcal{H}_2$-norm as:

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

Table 3: Computed 2-Norms

Computation	2-Norms
Frequency response method	1.8785
State-space method	1.8785
MATLAB command `norm`	1.8785

Table 3 shows that the results coincide with those obtained using the MATLAB command `norm`.

2.2 ∞-Norm

Two different methods are employed:

1. *Frequency response method*: the infinity norm is obtained as:

$$\|G\|_\infty = \max_{1\leq k\leq N} \sigma_{\max}\left[G(j\omega_k)\right]$$

   where $\sigma_{\max}$ is the largest singular value of $G(j\omega_k)$.

2. *Bounded real lemma*: analogously to the SISO case, the infinity norm is determined by iteratively checking whether

$$H = \begin{bmatrix} A & \gamma^{-2}BB^T \\ -C^TC & -A^T \end{bmatrix}$$

   has eigenvalues on the imaginary axis.

Table 4: Computed ∞-Norms

Computation	∞-Norm
Frequency response method	2.156
Bounded real lemma	2.156
MATLAB command `norm`	2.156

Table 4 shows that the results match those obtained using the MATLAB command `norm`.

2.3 MATLAB code

```
%% 1.2 Norms of MIMO systems

clc
close all
clear all
A = [-8.0, 2.5, -2.5; 7.8, -3.2, 13.8; 0.1, -0.5, -5.7];
B = [1, 0; 1, -1; 0, -2];
C = [0, 0, 1; 1, -1, 0];
D = zeros(2);
Gss = ss(A, B, C, D);  % state-space
G = tf(Gss);

% 1.2.1 2-NORM
% 1. Frequency response method
omega = linspace(0,5e4,1e5);    % relevant interval
n = length(A);
N = length(omega);
G_jomega = zeros(size(C,1), size(B,2), N);  % allocation of 3 matrix
trace_GstarG = zeros(1, N);
```

```
for k = 1:N
    G_jomega(:,:,k) = C * ((1j*omega(k)*eye(n) - A) \ B) + D;
    trace_GstarG(k) = trace(G_jomega(:,:,k)'*G_jomega(:,:,k));
end

Norm7 = sqrt(1/pi*trapz(omega, trace_GstarG));
disp(['2-norm = ', num2str(Norm7)]);

% 2. State-space method
[A,B,C,D] = ssdata(G);
L = are(A', zeros(size(A)), B*B');
Norm8 = sqrt(trace(C*L*C'));
disp(['2-norm = ', num2str(Norm8)]);

% 3. Validate results with norm()
disp(['2-norm = ', num2str(norm(G, 2))]);

% 1.2.2 INFINITY-NORM
% 1. Frequency response method
sigma = zeros(1, N);
for k = 1:N
    sigma(k) = sqrt(max(eig(G_jomega(:,:,k)'*G_jomega(:,:,k))));
end
Norm9 = max(sigma);
disp(['Infinity-norm = ', num2str(Norm9)]);

% 2. Bounded real lemma (iterative bisection algorithm)
gamma_u = 10;
gamma_l = 1e-5;     % boundaries
eps = 1e-8;
flag = true;
niter = 0;

while flag
    if (gamma_u-gamma_l)/gamma_l < eps
        flag = false;
        Norm10 = mean([gamma_u, gamma_l]);
    else
    gamma = mean([gamma_u, gamma_l]);
    H = [A, gamma^(-2)*B*B'; -C'*C, -A'];
    if any(abs(real(eig(H)))<1e-3)
        gamma_l = gamma;
    else
        gamma_u = gamma;
    end
    end
    niter = niter + 1;
end
disp(['Infinity-norm = ', num2str(Norm10)]);

% 3. Validate results with norm()
disp(['Infinity-norm = ', num2str(norm(G, inf))]);
```

3 Uncertainty modeling

The goal is to convert parametric uncertainty, defined by a structured model set, into multiplicative uncertainty, corresponding to an unstructured one, for the following transfer function:

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

where $a = 2 \pm 1$, $b = 3 \pm 2$, and $c = 10 \pm 3$.
The following steps are performed:

- Definition of the uncertain parameters a, b, and c using `ureal`.
- Definition of the uncertain LTI system with a, b, and c, whose `step`, `bode`, and `nyquist` diagrams are used for comparison with the multiplicative model in Figures 3, 4, 5, 7, 8, 9, 11, 12, and 13.
- Use of `usample` to generate two model uncertainty sets for G, the first consisting of 20 samples and the second of 200.
- Use of `ucover` to convert the multimodel uncertainty into a multiplicative one, starting from both the previously defined sets and G_{nominal}, which is based on the nominal parameters $a = 2$, $b = 3$, and $c = 10$.

In the following sections, the uncertainty filters W_2 are plotted for a first-order (Figure 2), second-order (Figure 6), and third-order system (Figure 10). It is worth noting that the difference between the filters computed using 20 and 200 samples is smaller when the system order is lower, meaning that less information is required to approximate a less complex filter. Moreover, the second- and third-order filters are very similar and both provide a good representation of the system's uncertainty.

It should also be noted that the first-order filter is less consistent than the others, whereas the third-order one provides only a marginal improvement over the second-order filter, making the latter the preferred choice.

In the other figures, multimodel uncertainty is compared with the multiplicative one for 20 and 200 samples, considering different orders.

Bode and Nyquist diagrams, along with step responses, indicate similar behavior between multimodel and multiplicative uncertainty, with differences arising from both sampling approximations and modeling techniques. The similarity between the results consistently increases when transitioning from first- to second-order filters.

Regarding the impact of the number of samples on the different responses, there is generally a strong similarity between the results obtained with 20 and 200 samples, regardless of the order. This suggests that the identification method converges quickly and that 20 samples are sufficient for an accurate representation of the system's uncertainty.

In conclusion, it seems that the best compromise between complexity and precision is to use a second-order filter with 20 samples, as it provides an accurate representation of the system's uncertainty while minimizing computational and modeling complexities.

3.1 First order filter

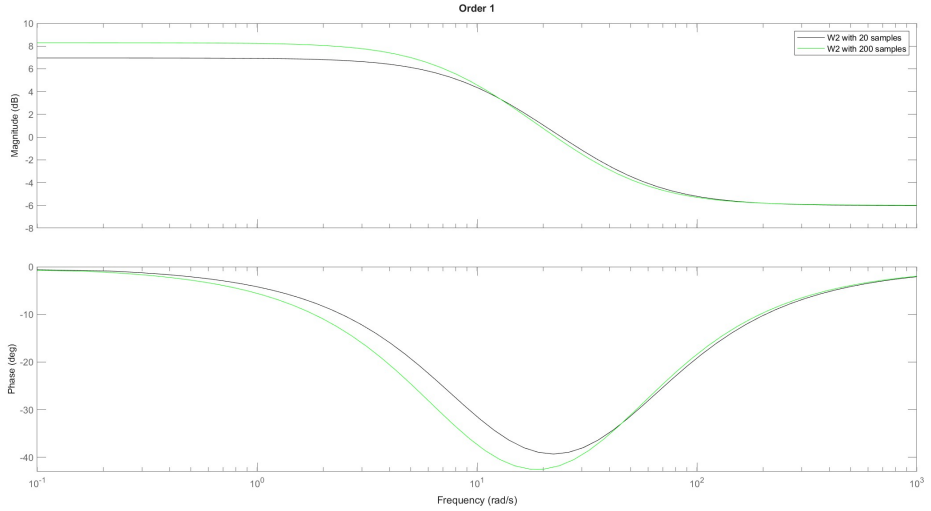

Figure 2: Bode response of uncertainty filters (1^{st} order)

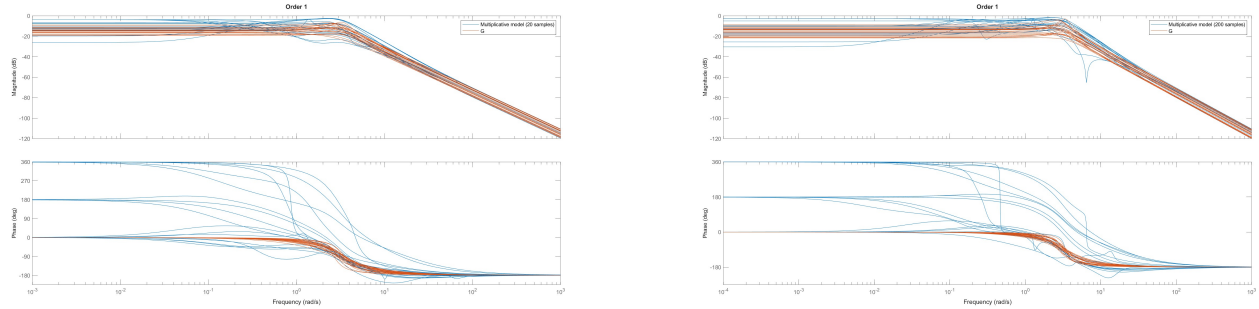

Figure 3: Bode response of models (1^{st} order) - 20 and 200 samples

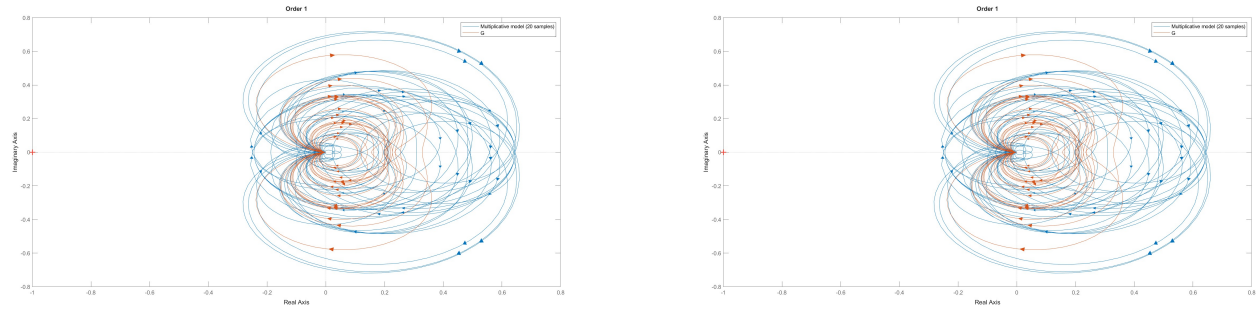

Figure 4: Nyquist response of models (1^{st} order) - 20 and 200 samples

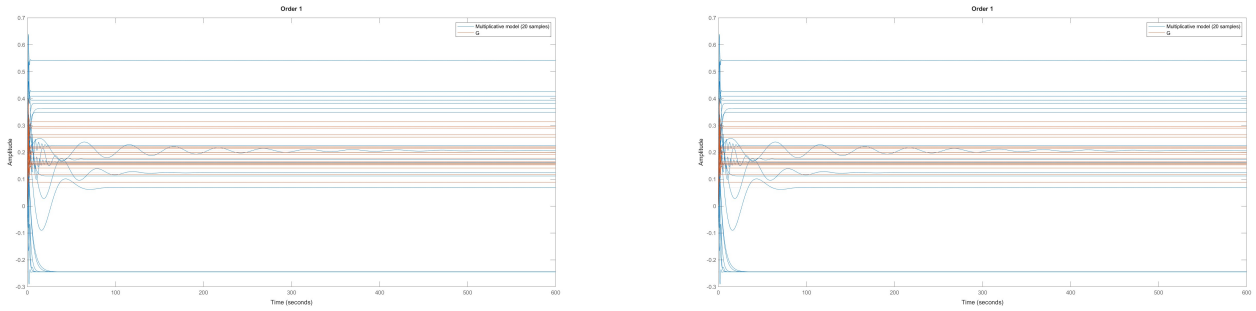

Figure 5: Step response of models (1^{st} order)- 20 and 200 samples

3.2 Second order filter

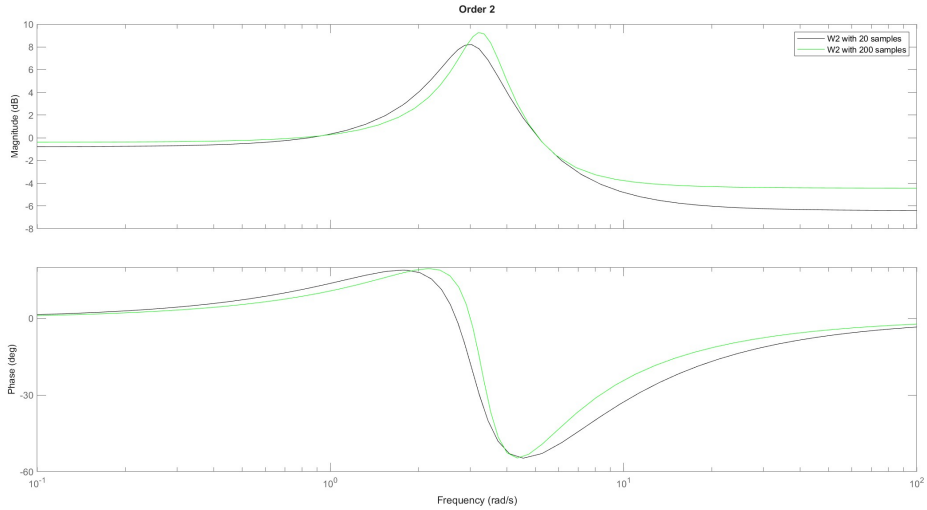

Figure 6: Bode response of uncertainty filters (2^{nd} order)

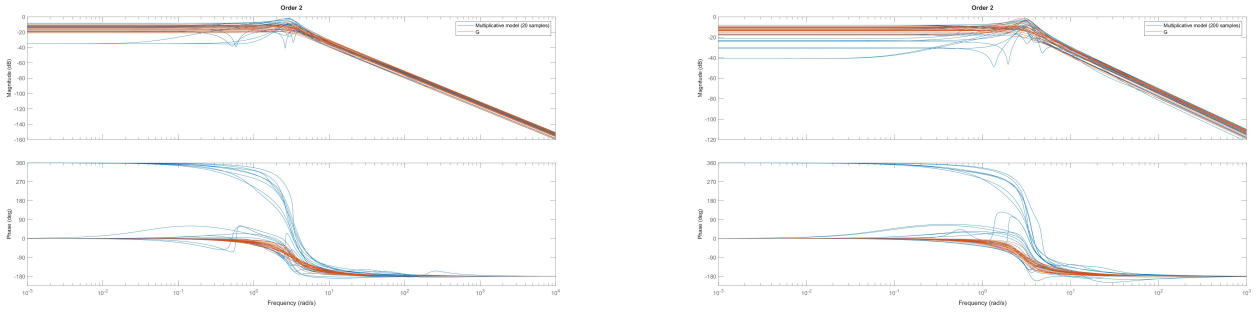

Figure 7: Bode response of models (2^{nd} order) - 20 and 200 samples

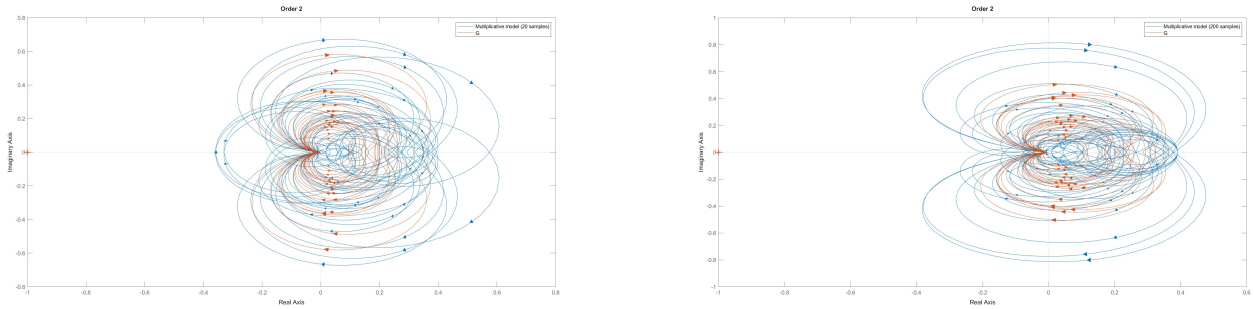

Figure 8: Nyquist response of models (2^{nd} order) - 20 and 200 samples

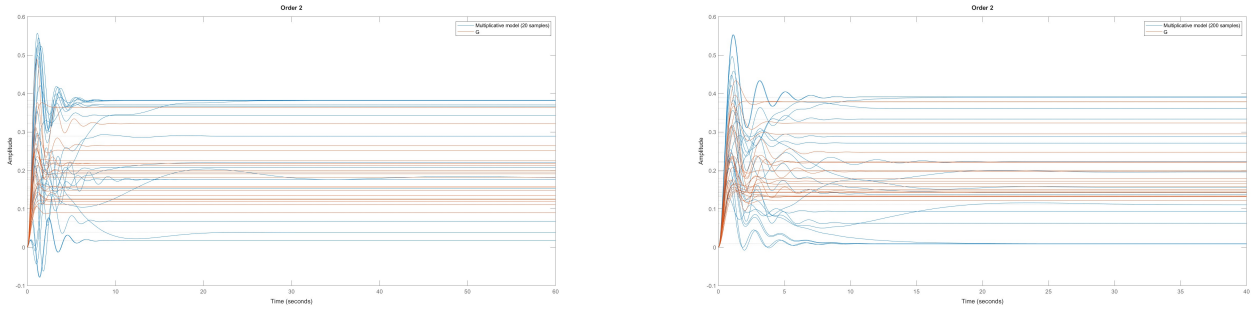

Figure 9: Step response of models (2^{nd} order) - 20 and 200 samples

3.3 Third order filter

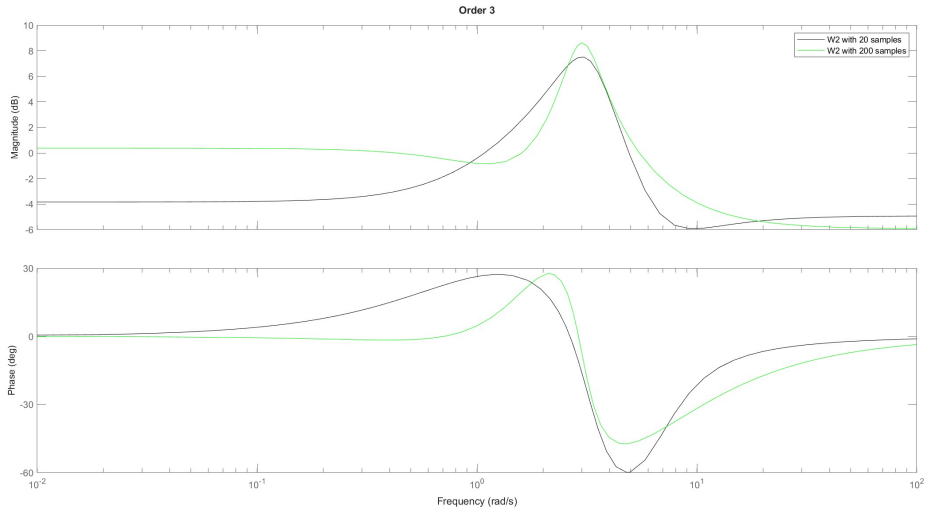

Figure 10: Bode response of uncertainty filters (3^{rd} order)

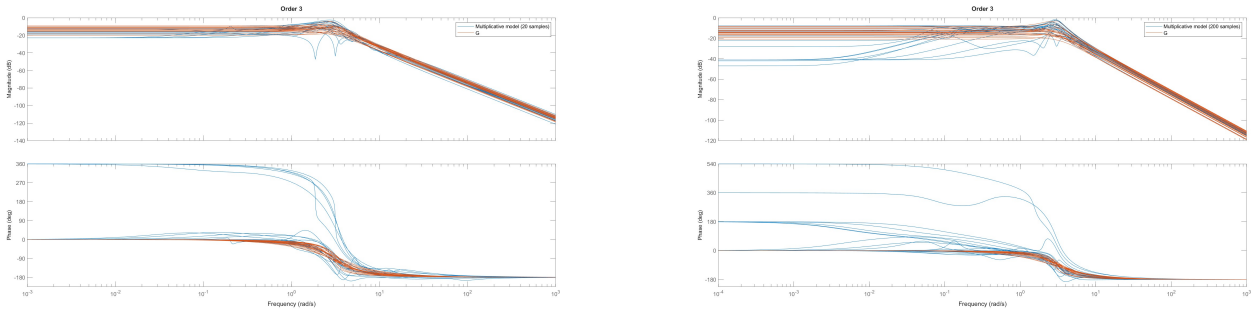

Figure 11: Bode response of models (3^{rd} order) - 20 and 200 samples

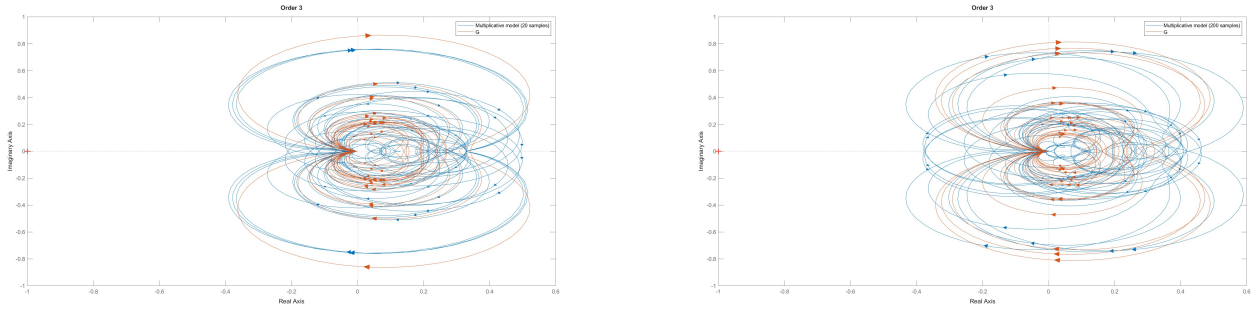

Figure 12: Nyquist response of models (3^{rd} order) - 20 and 200 samples

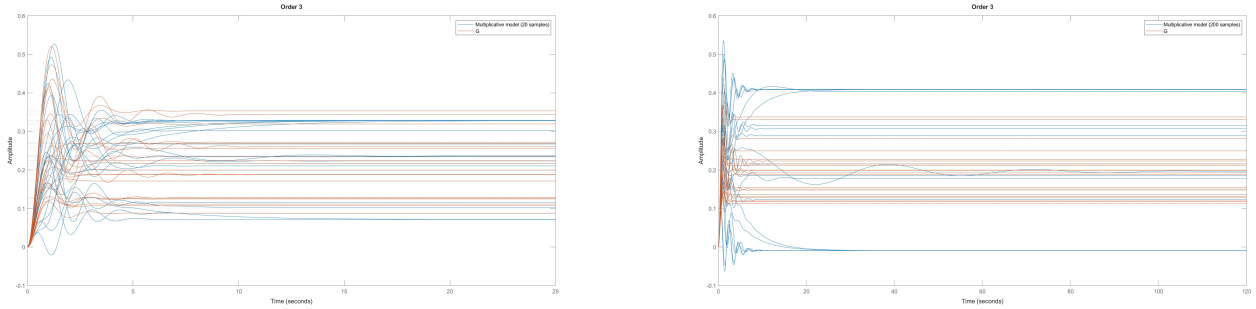

Figure 13: Step response of models (3^{rd} order) - 20 and 200 samples

3.4 MATLAB code

```
%% 1.3 Uncertainty modeling

clc
close all
clear all

% Parameters definition
a = ureal('a', 2, 'PlusMinus', 1);
b = ureal('b', 3, 'PlusMinus', 2);
c = ureal('c', 10, 'PlusMinus', 3);

% Define uncertain LTI system
```

```
num = a;
den = [1 b c];
G = tf(num, den);

% Use usample to generate two multimodel uncertainty sets (20, 200 samples)
sample1 = 20;
sample2 = 200;
G_samples1 = usample(G, sample1);
G_samples2 = usample(G, sample2);

% Creation of G_nominal (according to the nominal values of parameters)
G_nominal = tf(a.NominalValue, [1 b.NominalValue c.NominalValue]);

orders = 1:3; % Array of model orders to test

for order = orders

    % Estimate uncertainty models for the given order
    [usys1, info1] = ucover(G_samples1, G_nominal, order);
    [usys2, info2] = ucover(G_samples2, G_nominal, order);

    % Plot Bode response of uncertainty filters
    figure()
    bode(info1.W1, 'k', info2.W1, 'g');
    legend('W2 with 20 samples', 'W2 with 200 samples')
    title(['Order ', num2str(order)])
    set(gcf, 'Position', get(0, 'Screensize')); % Maximize figure
    saveas(gcf, fullfile('Images_CE1', ['Bode_Uncertainty_Order_', num2str(order), '.jpg']))

    % Plot Bode response of the multiplicative model (20 samples)
    figure()
    bode(usys1, G)
    legend('Multiplicative model (20 samples)', 'G')
    title(['Order ', num2str(order)])
    set(gcf, 'Position', get(0, 'Screensize'));
    saveas(gcf, fullfile('Images_CE1', ['Bode_Multiplicative_20_Order_', num2str(order), '.jpg']))

    % Plot Bode response of the multiplicative model (200 samples)
    figure()
    bode(usys2, G)
    legend('Multiplicative model (200 samples)', 'G')
    title(['Order ', num2str(order)])
    set(gcf, 'Position', get(0, 'Screensize'));
    saveas(gcf, fullfile('Images_CE1', ['Bode_Multiplicative_200_Order_', num2str(order), '.jpg']))

    % Plot step response for 20 samples
    figure()
    step(usys1, G)
    legend('Multiplicative model (20 samples)', 'G')
    title(['Order ', num2str(order)])
    set(gcf, 'Position', get(0, 'Screensize'));
    saveas(gcf, fullfile('Images_CE1', ['Step_20_Order_', num2str(order), '.jpg']))

    % Plot step response for 200 samples
    figure()
    step(usys2, G)
```

```
    legend('Multiplicative model (200 samples)', 'G')
    title(['Order ', num2str(order)])
    set(gcf, 'Position', get(0, 'Screensize'));
    saveas(gcf, fullfile('Images_CE1', ['Step_200_Order_', num2str(order), '.jpg']))

    % Plot Nyquist diagram for 20 samples
    figure()
    nyquist(usys1, G)
    legend('Multiplicative model (20 samples)', 'G')
    title(['Order ', num2str(order)])
    set(gcf, 'Position', get(0, 'Screensize'));
    saveas(gcf, fullfile('Images_CE1', ['Nyquist_20_Order_', num2str(order), '.jpg']))

    % Plot Nyquist diagram for 200 samples
    figure()
    nyquist(usys2, G)
    legend('Multiplicative model (200 samples)', 'G')
    title(['Order ', num2str(order)])
    set(gcf, 'Position', get(0, 'Screensize'));
    saveas(gcf, fullfile('Images_CE1', ['Nyquist_200_Order_', num2str(order), '.jpg']))
end
```
