CE2: Robust control of an electro-mechanical system

Advanced Control Systems

Computer Exercise 2

Gabelli Léo	403831
Vignoli Lorenzo	405238

May 5, 2025

Contents

Chapter 1

MULTIPLICATIVE UNCERTAINTY

This analysis focuses on the position control of a DC motor with a flexible element attached on top, resulting in large resonant modes at high frequencies (Figure 1.1). To create multimodel uncertainty, different weights are attached to the flexible elements at different positions.

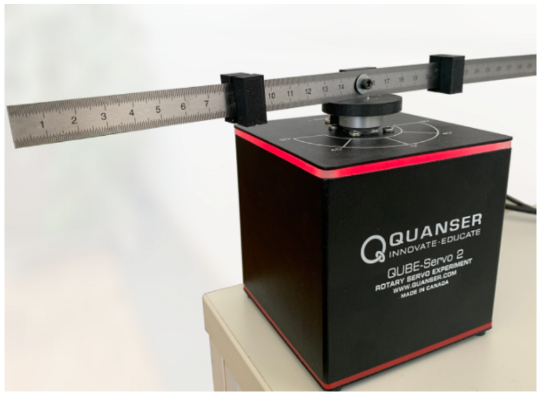


Figure 1.1: Quanser Servo-Qube

For identification purposes, the system is excited with a PRBS signal, and data are acquired with a sampling period of $T_s = 0.002\,\mathrm{s}$. The multimodel uncertainty is then converted into multiplicative uncertainty according to

$$\tilde{G} = G(1 + \Delta(s)W_2(s)),$$

where $G(s)$ is the nominal model of $\tilde{G}(s)$ and $\|\Delta\|_\infty \leq 1$.

1.1 Parametric and non-parametric model

Given the set of measurements under different loading conditions, both nonparametric and parametric models are identified. The former are obtained using the *Spectral Analysis* method (`spa`) applied to the detrended input-output data, estimating the frequency response function over the frequency grid. For the latter, the derivative of the output signal is first computed by applying a discrete-time differentiation operator, corresponding to $1 - z^{-1}$. An *Output Error* (`oe`) model is then identified from the differentiated data, with model orders set to $[10, 10, 1]$, corresponding respectively to the numerator order, denominator order, and input delay. Finally, the identified model is integrated from the derivative, and the procedure is repeated for all four experiments.

```
%% 1.1.1. Gf and G creation

clear, clc, close all
load('data_position.mat')

% Number of experiments and structure initialization
N_exp = 4;
Gf_struct(N_exp) = struct('Gf', []);
G_struct(N_exp) = struct('G', []);
G_der_struct(N_exp) = struct('G_der', []);

% Frequency
freqs = (pi/4096:pi/4096:pi)/Ts;

for ii=1:N_exp

    % Data extraction
    experiment_name = data{ii}.name;
    y = data{ii}.y;
    u = data{ii}.u;
    Z = detrend(iddata(y, u, Ts, "Period", 8191));

    % Frequency response of the system
    Gf_struct(ii).Gf = spa(Z, 8191, freqs);

    % Compute output derivative
    y_derivative = lsim(1 - tf('z',Ts)^-1, y);
    Z_der = detrend(iddata(y_derivative, u, Ts, "Period", 8191));

    % Use Output Error method to compute the model
    z = tf('z', Ts);
    G_der_struct(ii).G_der = oe(Z_der, [10, 10, 1]);    % order (x2), shift

    % Add back the integrator
    G_struct(ii).G = G_der_struct(ii).G_der / (1 - z^-1);

end

save('Model.mat')
```

1.2 Non-parametric frequency-domain uncertainty

For the model identified using spectral analysis, the frequency-domain uncertainty due to measurement noise can be visualized in the Nyquist diagram with a 95% confidence level.

The estimated $Re(\hat{G}(e^{j\omega}))$ and $Im(\hat{G}(e^{j\omega}))$ are asymptotically uncorrelated and normally distributed with equal variance $\Phi_v/2\Phi_u(\omega)$. As a consequence, the uncertainty regions of the non-parametric models are circular. Figure 1.2 shows the Nyquist diagrams of the four models, highlighting the differences among them.

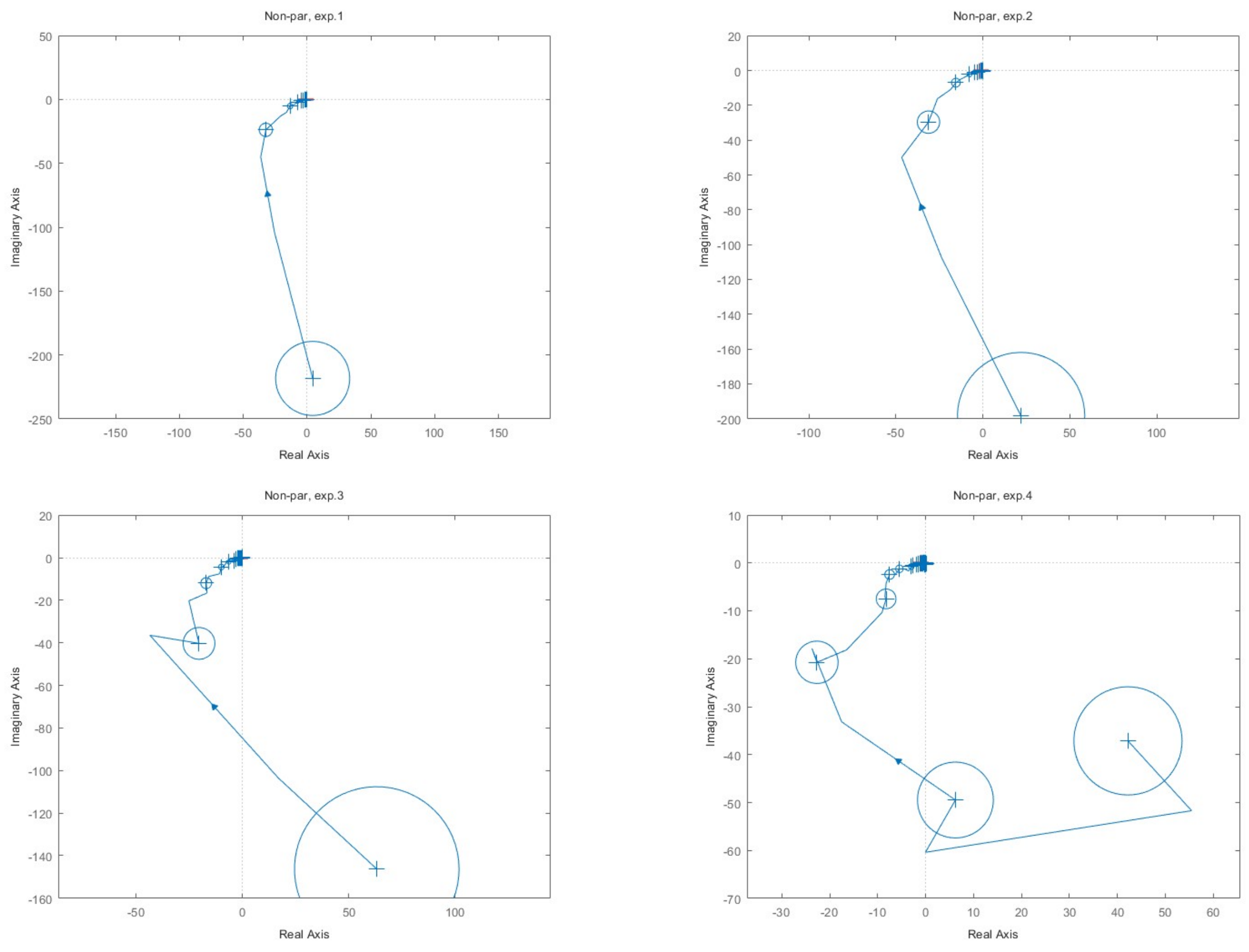


Figure 1.2: Non-parametric uncertainty

In Figure 1.3, the Bode responses of all nonparametric models can be observed. It is noticeable how the absence of parametric modeling makes the plots noisy and irregular.

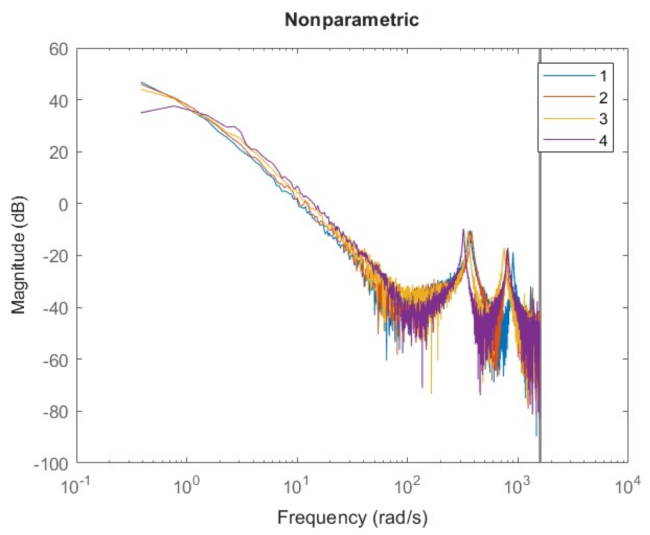
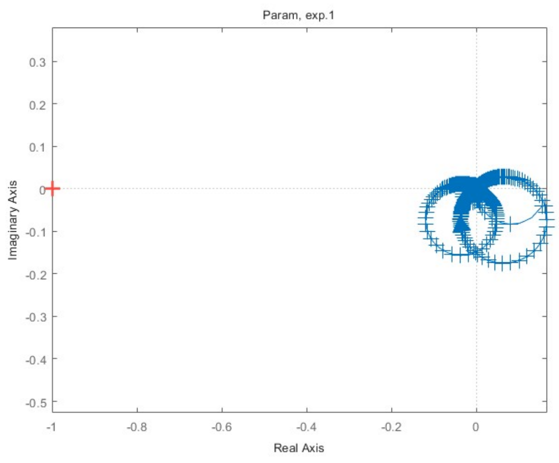


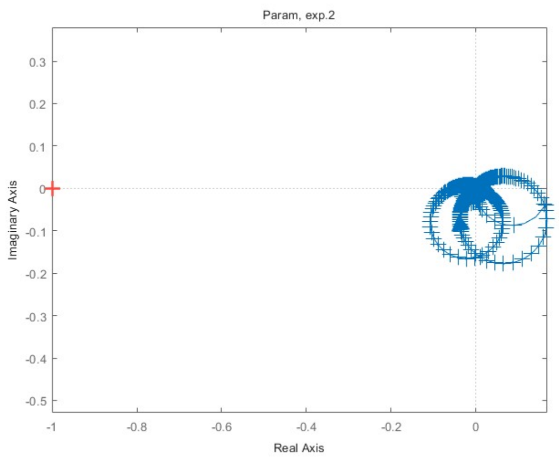
Figure 1.3: Bode response of non-parametric models

```
%% 1.1.2. Non-parametric models

clear, clc, close all
load('Model.mat')

% Nyquist options
opts = nyquistoptions;
opts.ConfidenceRegionDisplaySpacing = 3;
opts.ShowFullContour = 'off';

for ii=1:N_exp
    figure()
    nyquistplot(Gf_struct(ii).Gf, freqs, opts, 'sd', 2.45)
    hold on
    axis equal
    title(['Non-par, exp.', num2str(ii)])
end

% Nonparametric
figure()
bodemag(Gf_struct(1).Gf)
hold on
bodemag(Gf_struct(2).Gf)
bodemag(Gf_struct(3).Gf)
bodemag(Gf_struct(4).Gf)
legend('1', '2', '3', '4')
title('Nonparametric')
```

1.3 Parametric frequency-domain uncertainty

The frequency-domain uncertainty can be visualized in the Nyquist diagram with a 95% confidence level, as shown in Figure 1.5. The uncertainty regions are elliptical (see Figure 1.7), reflecting the multivariate Gaussian nature of the uncertainty, with a non-diagonal covariance matrix.





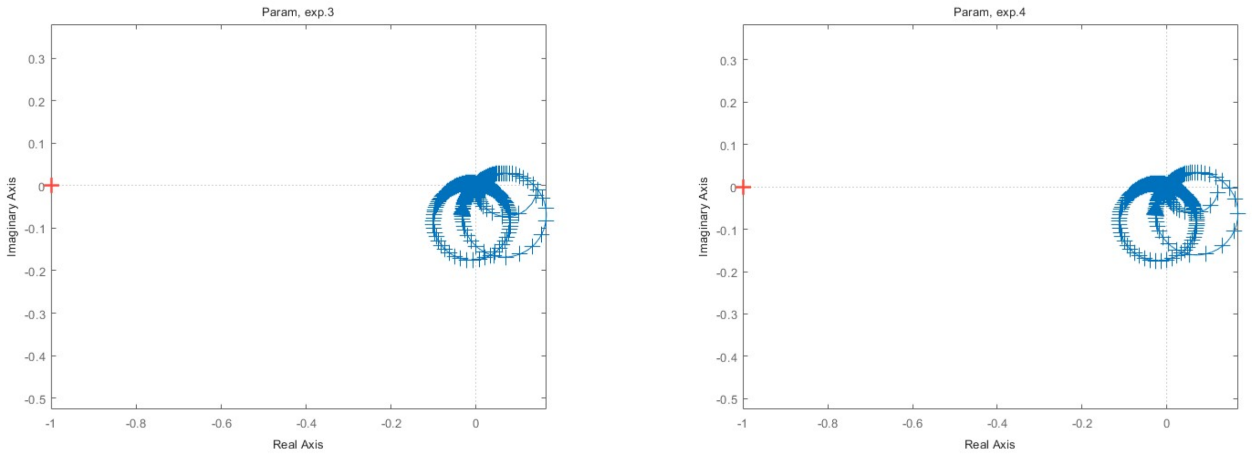


Figure 1.5: Parametric uncertainty

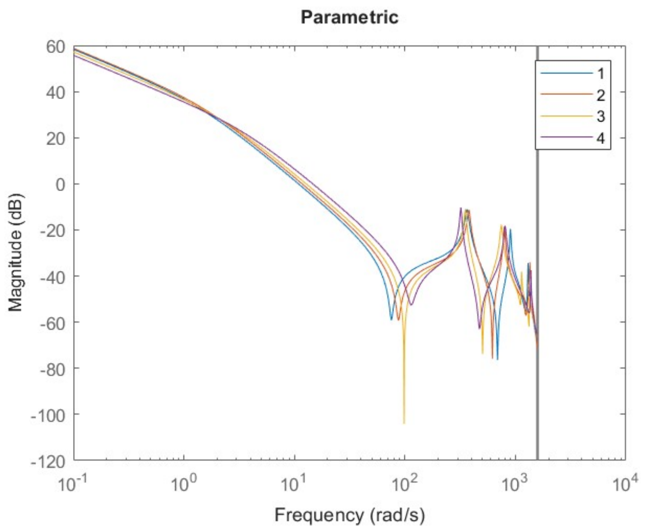


Figure 1.6: Bode response of parametric models

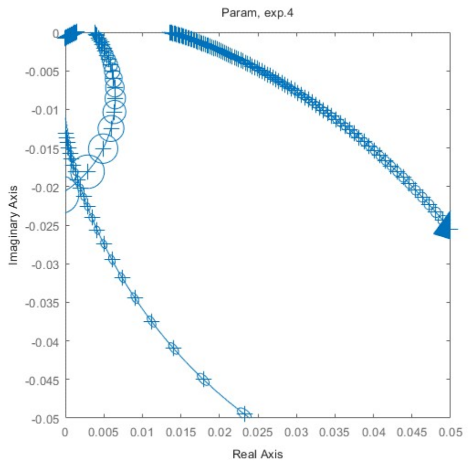


Figure 1.7: Detail of Nyquist plot of model 4

In Figure 1.6, the Bode responses of all parametric models can be observed. Parametric models exhibit a smoother shape compared to the nonparametric ones.

Observing the Nyquist plots in Figures 1.5 and 1.7, it can be noted that the uncertainty regions are much smaller, suggesting a reduced precision in the estimates derived from the nonparametric approach.

```
%% 1.1.3. Parametric models

clear, clc, close all
load('Model.mat')

% Nyquist options
opts = nyquistoptions;
opts.ConfidenceRegionDisplaySpacing = 3;
opts.ShowFullContour = 'off';

for ii=1:N_exp
    figure()
    nyquistplot(G_der_struct(ii).G_der, freqs, opts, 'sd', 2.45)
    axis equal
    hold on
    title(['Param, exp.', num2str(ii)])
end

% Parametric
figure()
bodemag(G_struct(1).G)
hold on
bodemag(G_struct(2).G)
bodemag(G_struct(3).G)
bodemag(G_struct(4).G)
legend('1', '2', '3', '4')
title('Parametric')
```

1.4 Multimodel set of parametric models

Since the uncertainty regions of the parametric models are smaller compared to those of the nonparametric ones, the parametric models are selected to represent the multimodel uncertainty, which can be expressed as $\mathcal{G} = \{G_1, G_2, G_3, G_4\}$.

```
clear, clc, close all
load('Model.mat')

Gmm = stack(1, G_struct(1).G, G_struct(2).G, G_struct(3).G, ...
    G_struct(4).G);
```

1.5 Weighting filter $\mathcal{W}_2$

By choosing a nominal model, the weighting filter W_2 that converts the multimodel uncertainty into multiplicative uncertainty is computed. In this case, W_2 is obtained from the derivative system models rather than from the plant models, to ensure greater consistency with the data processing approach. This equivalence holds because:

$$\frac{G_i}{G_{\text{nom}}} - 1 = \frac{G_{i,\text{der}}}{G_{\text{nom,der}}} - 1$$

since the integrator has been added *a posteriori*.

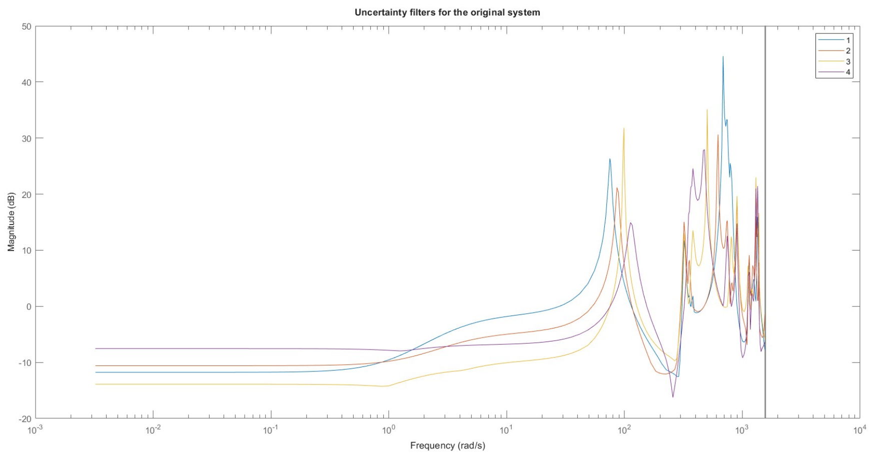


Figure 1.8: W_2 for different nominal models.

Figure 1.8 shows all the uncertainty filters for the system, obtained by imposing a different G_{nom} for each of the four possibilities. Figure 1.9 provides a comparison between the chosen filter W_2 (see Section 1.6) and its rational fixed-order approximation of order 20, which is selected for the following steps. The approximated filter is shown in blue, while the original one is shown in orange.

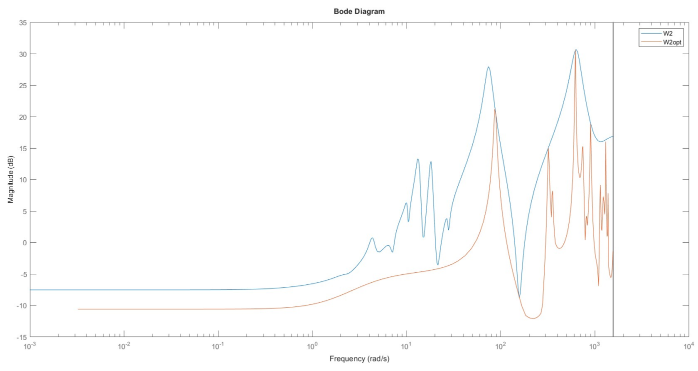


Figure 1.9: Rational fixed-order filter approximation

```
%% 1.1.5. W2 filter

clear, clc, close all
load('Model.mat')

Gmm = stack(1, G_struct(1).G, G_struct(2).G, G_struct(3).G, ...
    G_struct(4).G);

% Order 20 of filter
```

```
order = 20;

% Computation for every one
info = struct('info', [], 'W2', []);
for ii=1:4
    Gnom = G_struct(ii).G;
    [Gu, info(ii).info] = ucover(Gmm, Gnom, order, 'InputMult');
    info(ii).W2 = info(ii).info.W1opt;
end

% Plotting on bodemag
figure()
for ii = 1:4
    bodemag(info(ii).W2)
    hold on
end
legend('1', '2', '3', '4')
title('Uncertainty filters for the original system')

% We choose the second one because at low frequencies they are more or less
% the same, while 1 and 3 are too big at high frequencies and the
% fourth one has too big bandwidth in high frequencies.

% Plot the filter
figure()
bodemag(info(2).info.W1, info(2).info.W1opt)
legend('W2', 'W2opt')

% Saving information
W2 = info(2).info.W1;
G_nom = G_struct(2).G;
save('W2.mat', 'W2')
save('G_functions.mat', 'G_nom', 'G_struct', 'Gf_struct')
```

1.6 Best nominal model $\mathcal{G}_{nom}$

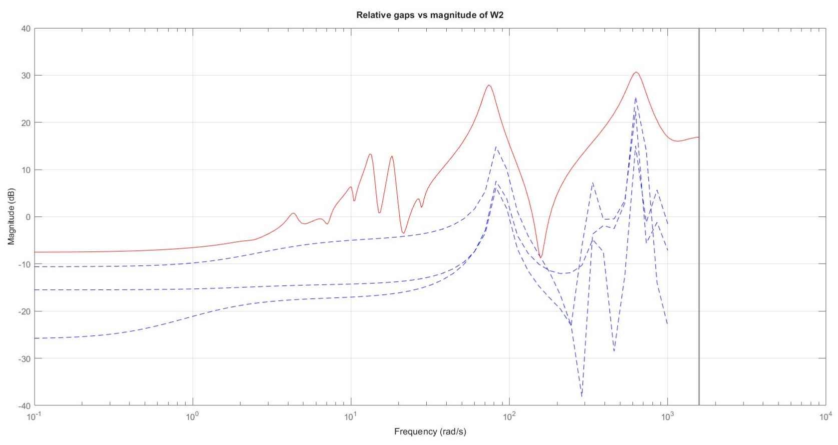


Figure 1.10: W_2 when G_2 is chosen as nominal system

Based on Figure 1.8, G_{nom} is chosen as the one whose resulting uncertainty filter provides the best performance. At low frequencies, all filters are comparable; however, at high frequencies, G_1 and G_3 result in excessively large amplitudes, while G_4 has an excessively large bandwidth. As a consequence, G_2 is selected as the nominal model.

Figure 1.10 shows the resulting W_2 (in red), compared to $\frac{G_i}{G_{\text{nom}}} - 1$. Overall, the latter remains below W_2, in accordance with the definition $\left|\frac{\tilde{G}(j\omega)}{G_{\text{nom}}(j\omega)} - 1\right| \leq |W_2(j\omega)|$; small overlaps are negligible and attributed solely to sampling errors.

```
%% 1.1.6. Comparison with original

clear, clc, close all
load('W2.mat')
load('G_functions.mat')

Gmm = stack(1, G_struct(1).G, G_struct(2).G, ...
    G_struct(3).G, G_struct(4).G);

figure()
hold on
Gf = frd(Gmm, logspace(-1,3,60));
bodemag((G_nom - Gf)/G_nom, 'b--', W2, 'r')
title('Relative gaps vs magnitude of W2')
grid on
```

Chapter 2

MODEL-BASED $\mathcal{H}_\infty$ CONTROL DESIGN

The aim of this analysis is to design a discrete-time $\mathcal{H}_\infty$ controller, initially minimizing the ∞-norm of the transfer functions from both the reference and the input disturbance to the output, and subsequently limiting the control input as well. In this and the following sections, the nominal model used is G_2.

2.1 Weighting filter $\mathcal{W}_1$

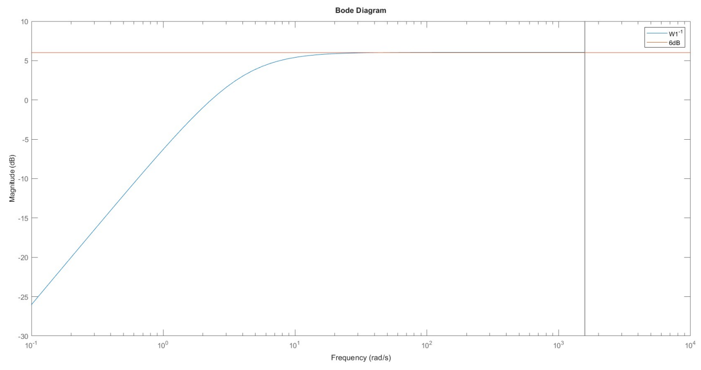


Figure 2.1: W_1

First, a discrete-time weighting filter W_1 is designed according to the following properties:

- Zero steady-state tracking error for a step reference, requiring integral action in the controller. To achieve this, the filter itself includes an integrator, implemented through a quasi-integrator.
- A modulus margin — defined as the minimum distance between the Nyquist plot and $(-1, 0)$ — of at least 0.5. Therefore, the high-frequency gain of W_1^{-1} is set to 2.
- The shortest settling time for the nominal model, achieved by choosing a numerator $s + 4$ in the continuous-time filter to make the system response more aggressive.

The resulting filter is:

$$W_1(z) = \frac{0.5z - 0.4960}{z - 1}$$

Figure 2.1 shows the Bode plot of the filter inverse W_1^{-1}, highlighting how W_1 satisfies the modulus margin condition by plotting a constant transfer function with a value of 2 (corresponding to 6 dB).

```
%% 2.2.1. W1

clear, clc, close all
load('Model.mat')

num = [1 4];          % settling time
den = [1 1e-5];       % quasi-integrator
W1 = tf(num, den)*1/2;
W1 = c2d(W1, Ts, 'zoh');

disp(W1)

% Checking behavior in high frequencies
figure()
bodemag(W1^-1, tf(2))
legend('W1^{-1}', '6dB')

save('W1.mat', 'W1')
```

2.2 Mixed sensitivity approach

The objective of this section is to design the controller K_∞ for robust performance. The mixed sensitivity approach is adopted, using the MATLAB function `mixsyn`. This function provides a controller that minimizes the $\mathcal{H}_\infty$ norm of the weighted closed-loop transfer function:

$$\begin{bmatrix} W_1 S \\ W_3 U \\ W_2 T \end{bmatrix}$$

where:

$$S = \frac{1}{1+GK}, \quad U = KS = \frac{K}{1+GK}, \quad T = 1 - S = \frac{GK}{1+GK}$$

In this application, the performance weighting filter W_1 is the one defined in Section 2.1, while W_3, acting on the input sensitivity function, is set to zero, and the uncertainty filter W_2 is the one defined in Section 1.6.

```
%% 2.2.2. Mixed sensitivity approach

clear, clc, close all
load('Model.mat')
load('W2.mat')
load('W1.mat')
load('G_functions.mat')

% Multimodel
Gmm = stack(1, G_struct(1).G, G_struct(2).G, ...
    G_struct(3).G, G_struct(4).G);
Kinf = mixsyn(G_nom, W1, [], W2);
```

2.3 Validation plots

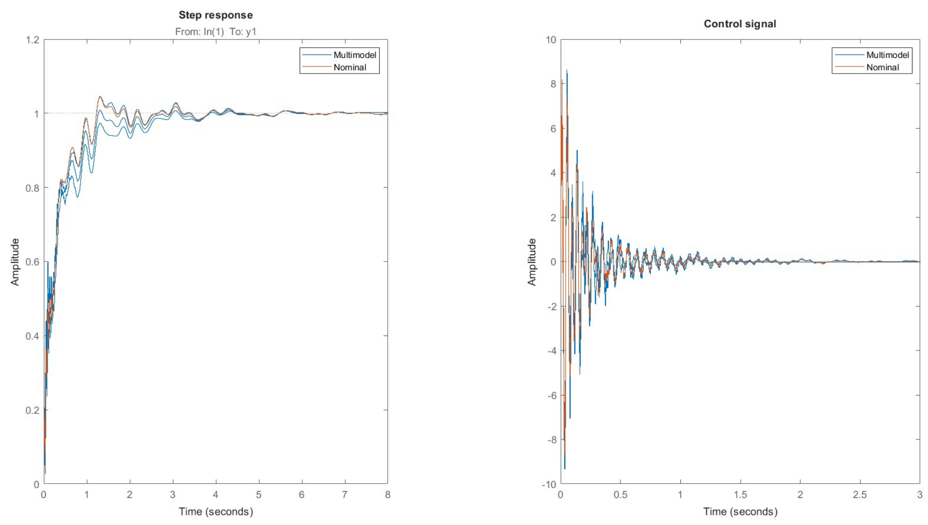


Figure 2.2: Step responses of closed-loop system and control signal

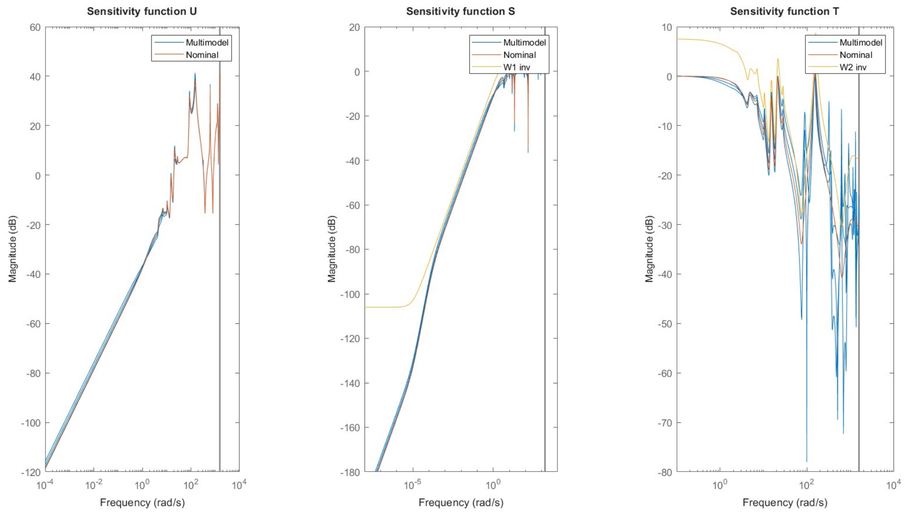


Figure 2.3: Bode of $\mathcal{U}$, $\mathcal{S}$, and $\mathcal{T}$

To validate the results, Figure 2.2 shows the step responses of the closed-loop system and of the control signal, while Figure 2.3 presents the magnitudes of the input sensitivity function $\mathcal{U}(z)$, the sensitivity function $\mathcal{S}(z)$, and the complementary sensitivity function $\mathcal{T}(z)$. The transfer functions $\mathcal{U}(z)$, $\mathcal{S}(z)$, and $\mathcal{T}(z)$ are computed using the MATLAB function `feedback`. Noticeably, the Bode plot of the input sensitivity function shows high magnitudes at high frequencies, suggesting that a limitation of the control input may be required.

The robust stability condition

$$\|W_2 T_{\text{nom}}\|_\infty < 1$$

is satisfied, since $\|W_2 T_{\text{nom}}\|_\infty = 0.7013$. Analogously, the robust-performance condition

$$\left\| \begin{bmatrix} W_1 S_{\text{nom}} \\ W_2 T_{\text{nom}} \end{bmatrix} \right\|_\infty < \frac{1}{\sqrt{2}}$$

is met, because

$$\left\| \begin{bmatrix} W_1 S_{\text{nom}} \\ W_2 T_{\text{nom}} \end{bmatrix} \right\|_\infty = 0.7034.$$

Table 2.1 reports the infinity norm of $W_1 S$, showing that for all systems the nominal performance condition $\|W_1 S\|_\infty < 1$ is satisfied (as visible in Figure 2.3). It is important to note that it is meaningless to evaluate the norm $\|W_2 T\|_\infty$ for systems other than the nominal one, because W_2 inherently represents the uncertainty of the multimodel set.

Table 2.1: $\|W_1 S\|_\infty$

G_i	Infinity Norm
G_1	0.6328
G_2	0.6028
G_3	0.6774
G_4	0.7999

```
%% 2.2.3. Validation plots and condition for robust performance

clear, clc, close all
load('Model.mat')
load('W2.mat')
load('W1.mat')
load('G_functions.mat')

Gmm = stack(1, G_struct(1).G, G_struct(2).G, ...
    G_struct(3).G, G_struct(4).G);
Kinf = mixsyn(G_nom, W1, [], W2);

Tnom = feedback(G_nom*Kinf, 1);
Snom = feedback(1, G_nom*Kinf);
Unom = feedback(Kinf, G_nom);
T = feedback(Gmm*Kinf, 1);
S = feedback(1, Gmm*Kinf);
U = feedback(Kinf, Gmm);

% Plot step response, control signal
figure()
subplot(1,2,1)
step(T,Tnom)
title('Step response')
legend('Multimodel','Nominal')
subplot(1,2,2)
step(U,Unom)
title('Control signal')
legend('Multimodel','Nominal')

figure()
```

```
subplot(1,3,1)
bodemag(U,Unom)
title('Sensitivity function U')
legend('Multimodel','Nominal')
subplot(1,3,2)
bodemag(S,Snom, W1^-1)
title('Sensitivity function S')
legend('Multimodel','Nominal', 'W1 inv')
subplot(1,3,3)
bodemag(T,Tnom,W2^-1)
title('Sensitivity function T')
legend('Multimodel','Nominal','W2 inv')

nominal_performance = norm(W1*S, inf);
disp('Nominal performance')
disp(nominal_performance)

robust_stability = norm(W2*Tnom, inf);
disp('Robust stability')
disp(robust_stability)

robust_performance = norm([W1*Snom, W2*Tnom], inf);
disp('Robust performance')
disp(robust_performance)
```

2.4 Weighting filter $\mathcal{W}_3$

The control signal $u(t)$ must remain within $\pm 5\,\mathrm{V}$ to avoid saturation during real-time implementation. Figure 2.2 shows that this specification is violated, motivating the introduction of a third weighting filter W_3. The value $W_3 = 0.05$ was selected by trial and error; the resulting responses are depicted in Figures 2.4 and 2.5. With this additional filter, the control signal stays below $\pm 5\,\mathrm{V}$.

The robust-stability condition is satisfied because

$$\|W_2 T_{\text{nom}}\|_\infty = 0.6912 < 1.$$

Similarly,

$$\left\| \begin{bmatrix} W_1 S_{\text{nom}} \\ W_2 T_{\text{nom}} \end{bmatrix} \right\|_\infty = 0.7126 \;\approx\; \frac{1}{\sqrt{2}},$$

which is only slightly larger than the value obtained in Section 2.3 due to the additional constraint introduced by W_3.

Table 2.2: $\|W_1 S\|_\infty$ - with W_3

G_i	Infinity Norm
G_1	0.6027
G_2	0.5750
G_3	0.6911
G_4	0.8161

Table 2.2 lists $\|W_1 S\|_\infty$ for all considered systems, confirming that the nominal performance requirement $\|W_1 S\|_\infty < 1$ is fulfilled. As in the case without W_3, the nominal system attains the smallest infinity norm.

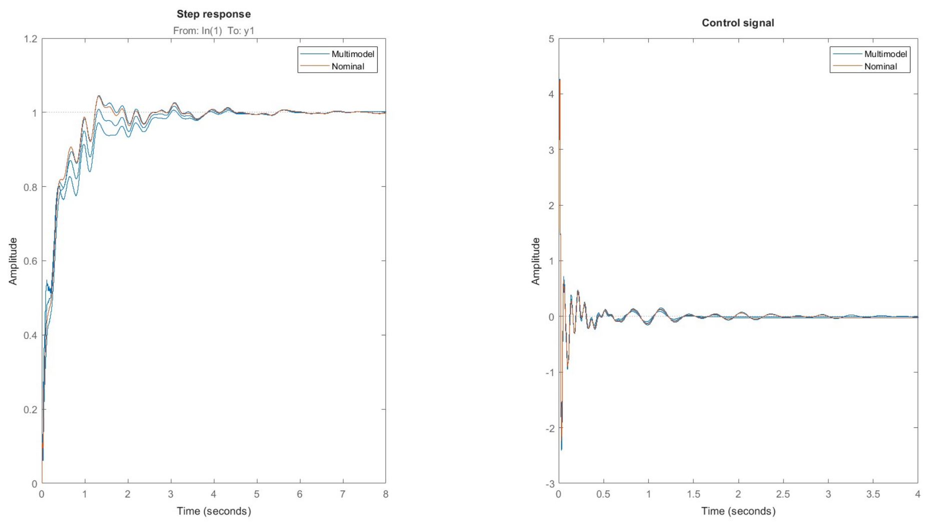


Figure 2.4: Step responses of closed-loop system and control signal - with W_3

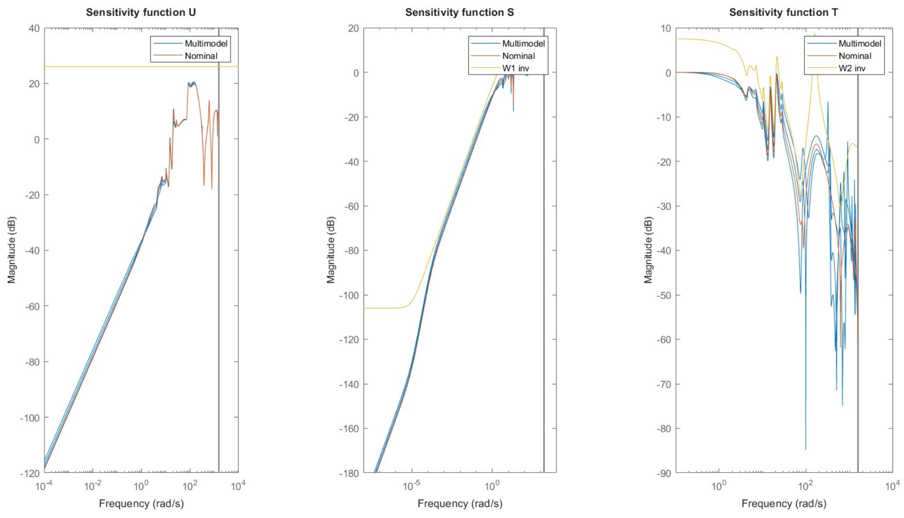


Figure 2.5: Bode of $\mathcal{U}$, $\mathcal{S}$, and $\mathcal{T}$ - with W_3

```
%% 2.2.4. W3

clear, clc, close all
load('Model.mat')
load('W2.mat')
load('W1.mat')
load('G_functions.mat')

Gmm = stack(1, G_struct(1).G, G_struct(2).G, ...
    G_struct(3).G, G_struct(4).G);

W3 = tf(0.05);      % trial and error
save('W3.mat', 'W3')

Kinf_3 = mixsyn(G_nom, W1, W3, W2);
```

```
Tnom = feedback(G_nom*Kinf_3, 1);
Snom = feedback(1, G_nom*Kinf_3);
Unom = feedback(Kinf_3, G_nom);
T = feedback(Gmm*Kinf_3, 1);
S = feedback(1, Gmm*Kinf_3);
U = feedback(Kinf_3, Gmm);

figure()
subplot(1,2,1)
step(T,Tnom)
title('Step response')
legend('Multimodel','Nominal')
subplot(1,2,2)
step(U,Unom)
title('Control signal')
legend('Multimodel','Nominal')

figure()
subplot(1,3,1)
bodemag(U,Unom,W3^-1)
title('Sensitivity function U')
legend('Multimodel','Nominal')
subplot(1,3,2)
bodemag(S,Snom, W1^-1)
title('Sensitivity function S')
legend('Multimodel','Nominal', 'W1 inv')
subplot(1,3,3)
bodemag(T,Tnom,W2^-1)
title('Sensitivity function T')
legend('Multimodel','Nominal','W2 inv')

nominal_performance = norm(W1*S, inf);
disp('Nominal performance')
disp(nominal_performance)

robust_stability = norm(W2*Tnom, inf);
disp('Robust stability')
disp(robust_stability)

robust_performance = norm([W1*Snom, W2*Tnom], inf);
disp('Robust performance')
disp(robust_performance)

save('H_inf_controller.mat')
```

2.5 Controller order reduction

The comparatively large order (32) of the initial controller originates from the high order (20) of the weighting filter W_2. This complexity results in some pole–zero cancellations, with poles and zeros located extremely close to one another. To simplify the problem, the MATLAB function `reduce` is applied, producing a 25-order controller. Figure 2.6 illustrates the effect: the pole–zero map of the full-order controller is shown on the *left*, whereas that of the reduced controller is on the *right*. The pole–zero cancellations near the origin disappear after reduction.

Regarding the closed-loop norms,

$$\|W_2 T_{\text{nom}}\|_\infty = 0.6912,$$

the same value obtained with the unreduced controller. By contrast,

$$\| \begin{bmatrix} W_1 S_{\text{nom}} \\ W_2 T_{\text{nom}} \end{bmatrix} \|_\infty = 0.7600,$$

which is higher than before, indicating that order reduction entails a modest loss in robust performance.

Table 2.3: $\|W_1 S\|_\infty$ - reduced K_∞

G_i	Infinity Norm
G_1	0.6190
G_2	0.5580
G_3	0.6384
G_4	0.7539

Nevertheless, Table 2.3 confirms that the nominal-performance requirement remains satisfied. Figures 2.7 and 2.8 show that the time-domain performance is still very close to that of the full-order K_∞, while Figure 2.9 compares the Bode diagrams of the two controllers, whose differences are negligible.

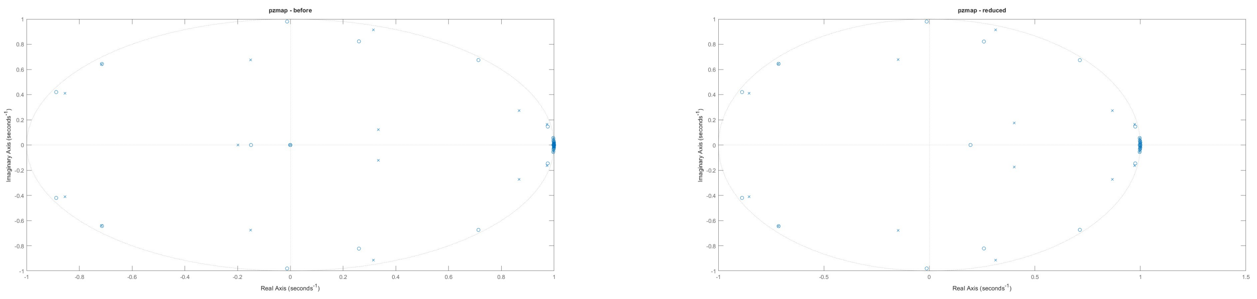


Figure 2.6: Maps of poles and zeros

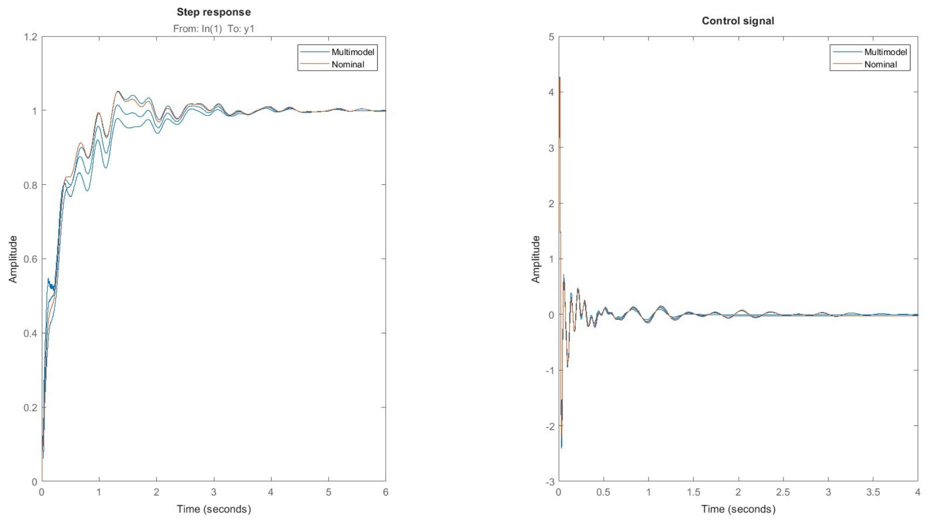


Figure 2.7: Step responses of closed-loop system and control signal - reduced K_∞

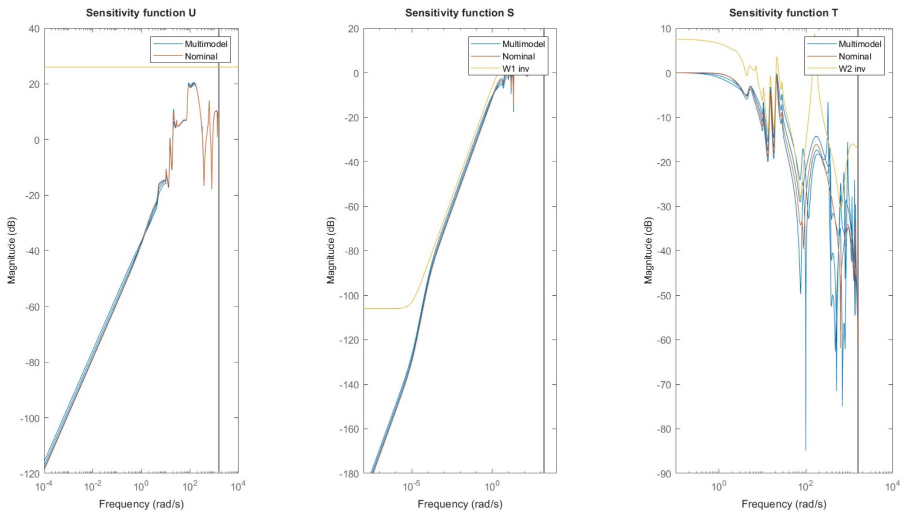


Figure 2.8: Bode of $\mathcal{U}$, $\mathcal{S}$, and $\mathcal{T}$ - reduced K_∞

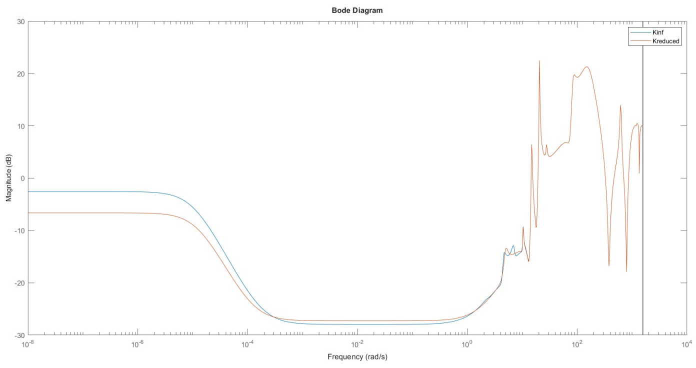


Figure 2.9: Bode diagram of K_∞ and $K_{\infty,reduced}$

```
%% 2.2.5. The order of the controller may be too large: pzmap.

clear, clc, close all
load('H_inf_controller.mat')

Gmm = stack(1, G_struct(1).G, G_struct(2).G, ...
    G_struct(3).G, G_struct(4).G);
Kinf = Kinf_3;

% Order of Kinf
orderK = size(Kinf.A,1);
disp(['Order of the controller: ', num2str(orderK)])

% High order of the controller, especially due to the high order of W2
figure()
pzmap(Kinf)

% Reduction
Kreduced = reduce(Kinf, 25);
orderK = size(Kreduced.A, 1);
```

```
disp(['Order of the controller: ', num2str(orderK)])
figure()
pzmap(Kreduced)

Tnom = feedback(G_nom*Kreduced, 1);
Snom = feedback(1, G_nom*Kreduced);
Unom = feedback(Kreduced, G_nom);
T = feedback(Gmm*Kreduced, 1);
S = feedback(1, Gmm*Kreduced);
U = feedback(Kreduced, Gmm);

figure()
subplot(1,2,1)
step(T,Tnom)
title('Step response')
legend('Multimodel','Nominal')
subplot(1,2,2)
step(U,Unom)
title('Control signal')
legend('Multimodel','Nominal')

figure()
subplot(1,3,1)
bodemag(U,Unom,W3^-1)
title('Sensitivity function U')
legend('Multimodel','Nominal')
subplot(1,3,2)
bodemag(S,Snom, W1^-1)
title('Sensitivity function S')
legend('Multimodel','Nominal', 'W1 inv')
subplot(1,3,3)
bodemag(T,Tnom,W2^-1)
title('Sensitivity function T')
legend('Multimodel','Nominal','W2 inv')

% Plot of the original controller and of the reduced one
figure()
bodemag(Kinf, Kreduced)
legend('Kinf', 'Kreduced')

nominal_performance = norm(W1*S, inf);
disp('Nominal performance')
disp(nominal_performance)

robust_stability = norm(W2*Tnom, inf);
disp('Robust stability')
disp(robust_stability)

robust_performance = norm([W1*Snom, W2*Tnom], inf);
disp('Robust performance')
disp(robust_performance)

save('Norm_validation.mat')
```

Chapter 3

MODEL-BASED $\mathcal{H}_2$ CONTROLLER DESIGN

The aim of this analysis is to design a state feedback controller such that the sum of the squared two-norms of the closed-loop transfer functions from the input disturbance to the output and to the system's input is minimized. The controller K_2 is computed by solving an SDP problem with `YALMIP`, and the result is compared with that of the Linear Quadratic Regulator (LQR).

3.1 Conversion to continuous time model

To obtain the controller, the system must first be converted from discrete-time to continuous-time form.

```
%% 3.3.1. Conversion to continuous time model

clc, clear, close all
load("Model.mat")

Gct = d2c(G_struct(2).G);
[A, B, C, D] = ssdata(Gct);
save('continuous_model.mat')
```

3.2 State-space equations of the closed-loop system

Consider the state-space representation of a strictly proper system

$$\begin{cases} \dot{x}(t) = Ax(t) + Bu(t), \\ y(t) = Cx(t) \end{cases}$$

With state-feedback control, the input signal is

$$u(t) = -Kx(t) + v(t)$$

where $v(t)$ is an input disturbance.

To minimize the two transfer functions, define

$$\begin{cases} y_1(t) = Cx(t), \\ y_2(t) = -Kx(t) \end{cases}$$

and obtain the closed-loop model

$$\begin{cases} \dot{x}(t) = (A - BK)x(t) + Bv(t), \\ y_1(t) = Cx(t), \\ y_2(t) = -Kx(t) \end{cases}$$

The sum of the squared norms of the two transfer functions can therefore be written as

$$\left\| \begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix} \right\|_2^2$$

3.3 Convex optimization problem

Employing the H_2 *norm by SDP* Lemma, the problem can be reformulated as

$$\begin{aligned} \min_{L} \quad & \operatorname{trace}\big(CLC^\mathsf{T}\big) \; + \; \operatorname{trace}\big(KLK^\mathsf{T}\big) \\ \text{s.t.} \quad & \big(A - BK\big)L + L\big(A - BK\big)^\mathsf{T} + BB^\mathsf{T} \; \preceq \; 0, \\ & L \succ 0. \end{aligned}$$

However, both the objective function and the constraint contain products of decision variables, so the formulation must be modified to obtain LMI objectives and constraints. A simple trick is to introduce a new variable $X = K\,L$, which gives the LMI

$$AL + LA^\mathsf{T} - BX - X^\mathsf{T}B^\mathsf{T} + BB^\mathsf{T} \; \preceq \; 0.$$

Since X and L now appear only linearly, the above inequality is an LMI. The second term in the trace, however, is KLK^T, so a new variable M can be defined such that

$$M \succeq KLK^\mathsf{T} \quad \iff \quad M - KLK^\mathsf{T} \succeq 0.$$

Using $LK^\mathsf{T} = X^\mathsf{T}$ and $K = XL^{-1}$, this becomes

$$M - XL^{-1}X^\mathsf{T} \succeq 0,$$

which, by Schur's Lemma, is equivalent to

$$\begin{bmatrix} M & X \\ X^\mathsf{T} & L \end{bmatrix} \succeq 0.$$

Hence, the optimization problem can be written as

$$\begin{aligned} \min_{L,\,X,\,M} \quad & \operatorname{tr}\big(CLC^\mathsf{T}\big) + \operatorname{tr}(M) \\ \text{s.t.} \quad & AL + LA^\mathsf{T} - BX - X^\mathsf{T}B^\mathsf{T} + BB^\mathsf{T} \; \preceq \; 0, \\ & \begin{bmatrix} M & X \\ X^\mathsf{T} & L \end{bmatrix} \succeq 0, \\ & L \succ 0. \end{aligned}$$

3.4 $\mathcal{H}_2$ controller

The controller is obtained by solving the optimization problem with MOSEK, and the state-feedback gain matrix is recovered as

$$K_{H_2} = XL^{-1}.$$

```
%% 3.3.4. H2 controller

clear, clc, close all
load('continuous_model.mat')

n = size(A, 1);
m = size(B, 2);

% Decision variables
L = sdpvar(n, n, 'symmetric');
X = sdpvar(m, n);
M = sdpvar(m, m, 'symmetric');

% Objective and LMIs
obj = trace(C*L*C') + trace(M);
lmi1 = A*L + L*A' - B*X - X'*B' + B*B' <= 0;
lmi2 = [M X; X' L] >= 0;
lmi3 = L >= eps;
lmi = [lmi1, lmi2, lmi3];

% MOSEK optimization
options = sdpsettings('solver', 'mosek');
optimize(lmi, obj, options);
K_H2 = value(X) * inv(value(L));
save('H_2_controller.mat', 'A', 'B', 'C', 'D', 'K_H2', 'm', 'n')
```

3.5 Step response of the closed-loop system

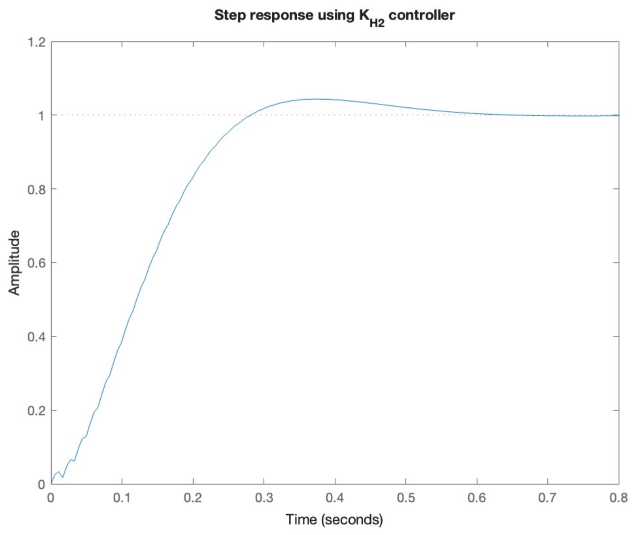


Figure 3.1: Step response of the H_2-controlled system

```
%% 3.3.5. Step response of the closed-loop system

clear, clc, close all
load('H_2_controller.mat')

% State-space representation
Acl = A - B*K_H2;
Bcl = B;
Ccl = C;
Dcl = D;
sys_closedloop = ss(Acl,Bcl,Ccl,Dcl);

figure()
step(sys_closedloop)
title('Step response using H2-controller')
save('H_2_system.mat')
```

3.6 Comparison with LQR

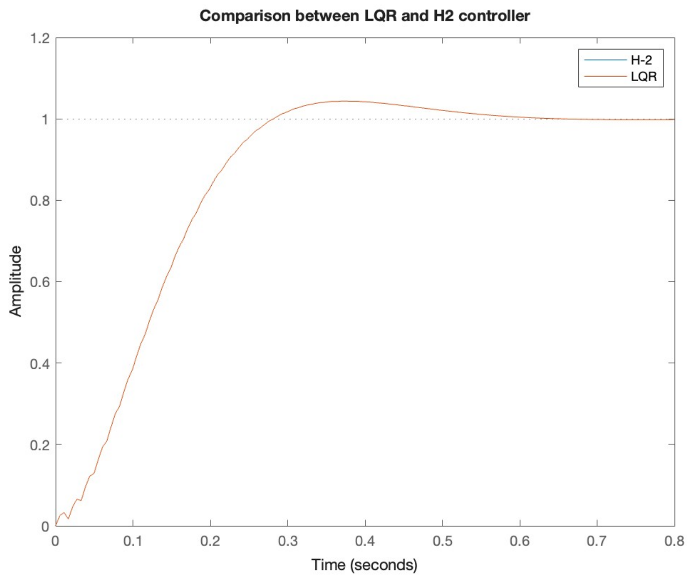


Figure 3.2: $\mathcal{H}_2$ vs. LQR.

For a continuous-time system, the Linear Quadratic Regulator (LQR) computes the state-feedback control $u = -K_{\text{LQR}} \cdot x$ that minimizes the quadratic cost function

$$J(u) = \int_0^\infty \left(x^\mathsf{T} Q x + u^\mathsf{T} R u\right) dt.$$

Because $y = Cx$, choosing $Q = C^\mathsf{T} C$ and $R = I$ gives

$$J(u) = \int_0^\infty \left(|y|^2 + |u|^2\right) dt = \int_0^\infty \left(|y_1|^2 + |y_2|^2\right) dt = \int_{-\infty}^\infty \left(|y_1|^2 + |y_2|^2\right) dt,$$

since $y_1(t) = y_2(t) = 0$ for $t < 0$.

Hence minimizing $J(u)$ is equivalent to minimizing

$$\left\| \begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix} \right\|_2^2,$$

which is exactly the same optimization problem as the H_2 formulation. Consequently, the step responses of the two controllers (shown in Figure 3.2) coincide.

```
%% 3.3.6. Comparison with LQR

clear, clc, close all
load('H_2_system.mat')

% LQR
Q = C'*C;              % Q (min. errors on state)
R = eye(m);            % R (min. errors on input)
[K_lqr, ~, ~] = lqr(A, B, Q, R);
Acl_lqr = A - B*K_lqr;
sys_closedloop_lqr = ss(Acl_lqr, Bcl, Ccl, Dcl);

figure()
step(sys_closedloop)
hold on
step(sys_closedloop_lqr)
title('Comparison between LQR and H2 controller')
legend('H-2', 'LQR')
```

Chapter 4

DATA-DRIVEN CONTROLLER

4.1 Multiplicative uncertainty

The aim of this analysis is to compute a $\mathcal{H}_\infty$ controller using the system's frequency response. A data-driven approach is adopted, implemented through the MATLAB library `datadriven`. The controller order is fixed at 5, with $K_c = 1$ serving as the initial stabilizing controller to ensure closed-loop stability. It should be noted that since the data-driven optimization does not inherently guarantee stability (unlike the Bounded Real Lemma approach), one must ensure controller stability throughout the entire process. The frequency grid is implemented as specified in the following code, and the performance filters remain the same as those used in Chapter 2.

```
%% 4.4.1. Data-driven controller: multiplicative uncertainty

clear, clc, close all
rng(22)
load("Model.mat")
load("G_functions.mat")
load('W1.mat')
load('W2.mat')
load('W3.mat')

Gmm = stack(1, G_struct(1).G, G_struct(2).G, ...
    G_struct(3).G, G_struct(4).G);
omegas = unique([ ...
    logspace(log10(0.4), log10(pi/Ts), 200), ...
    linspace(300, 400, 101), ...
    linspace(100, 1000, 101), ...
    ]);
```

4.1.1 Data-driven control

The Linear Fractional Transformation is used to define the augmented plant required by the `datadriven` library to solve the mixed-sensitivity problem. Figure 4.1 shows the step responses, demonstrating that the settling time is significantly longer compared to those observed in previous chapters. This slower response is most likely attributable to the low controller order, because achieving stability with a reduced-order controller inevitably compromises speed. Figure 4.2 presents instead the Bode diagrams of the sensitivity functions $\mathcal{U}$, $\mathcal{S}$, and $\mathcal{T}$.

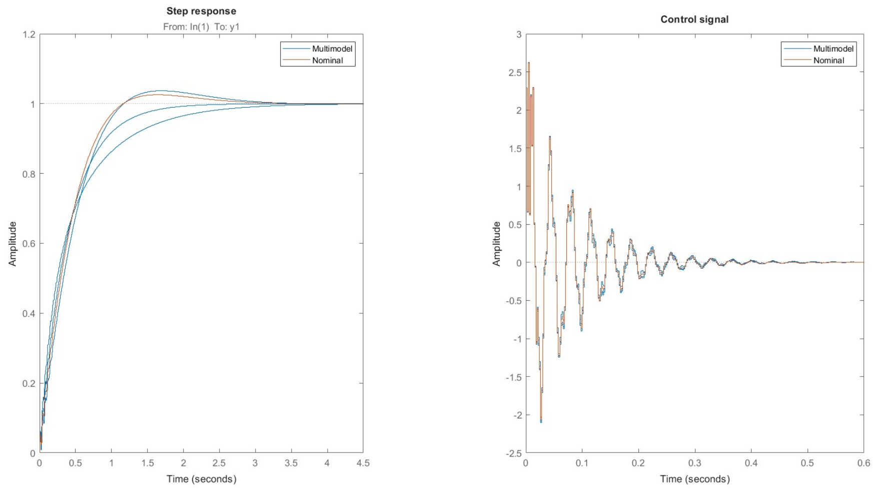


Figure 4.1: Step responses of closed-loop system and control signal - multiplicative

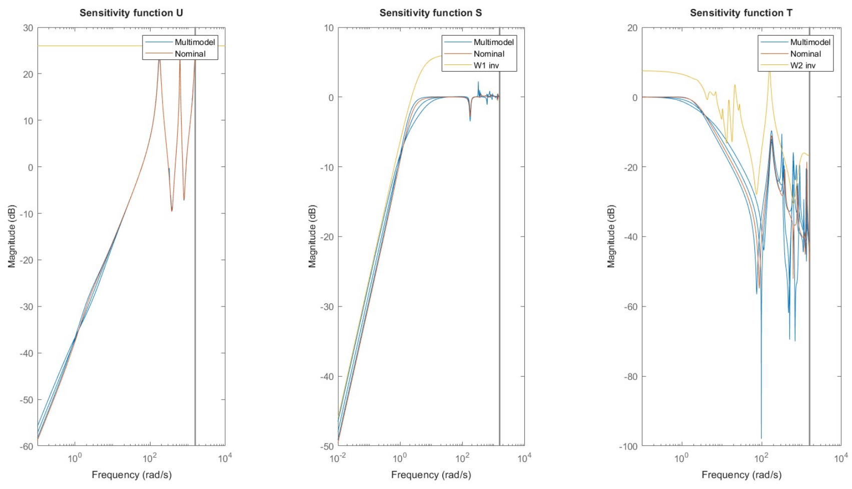


Figure 4.2: Bode of $\mathcal{U}$, $\mathcal{S}$, and $\mathcal{T}$ - multiplicative

```
% 1. Proportional initial controller
order = 5;
ny = 1;
nu = 1;
K_c = c2d(tf(1), Ts, 'zoh');

% Check that K_c is a stabilizing controller
CL_nom = feedback(1, K_c*G_nom);
isstable(CL_nom)

% LFR model
P = augw(G_nom, W1, W3, W2);
P = mktito(P, ny, nu);

% Problem formulation
K = datadriven.Controller.SS(order, ny, nu, Ts);
K.setinitial(K_c);
```

```
% Objective and ensuring stability (not needed in bounded real lemma)
synthesiser = Synthesiser(K);
synthesiser.add_Hinf_objective(P, omegas);
synthesiser.ensure_controller_stability(omegas);

% Output
output = synthesiser.synthesise();
K_final = output.Controller;

% Plotting
Tnom = feedback(G_nom*K_final, 1);
Snom = feedback(1, G_nom*K_final);
Unom = feedback(K_final, G_nom);
T = feedback(Gmm*K_final, 1);
S = feedback(1, Gmm*K_final);
U = feedback(K_final, Gmm);

figure()
subplot(1,2,1)
step(T,Tnom)
title('Step response')
legend('Multimodel','Nominal')
subplot(1,2,2)
step(U,Unom)
title('Control signal')
legend('Multimodel','Nominal')

figure()
subplot(1,3,1)
bodemag(U,Unom,W3^-1)
title('Sensitivity function U')
legend('Multimodel','Nominal')
subplot(1,3,2)
bodemag(S,Snom, W1^-1)
title('Sensitivity function S')
legend('Multimodel','Nominal', 'W1 inv')
subplot(1,3,3)
bodemag(T,Tnom,W2^-1)
title('Sensitivity function T')
legend('Multimodel','Nominal','W2 inv')
```

4.1.2 Comparison with model-based methods

One of the main advantages of the data-driven approach is its ability to design fixed and low-order controllers through convex optimization. In contrast, model-based methods are bound to the order of the augmented plant and therefore typically produce high-order controllers. Section 2.5 reduced the controller order to 25 to remove pole–zero cancellations; here the order is limited to 5 and the outcomes are compared in Figures 4.3 and 4.4. The resulting reduced-order model-based controller leaves the system unstable, showing that, although data-driven methods can successfully deliver fixed-order gains K, achieving the same with a purely model-based approach is often impossible.

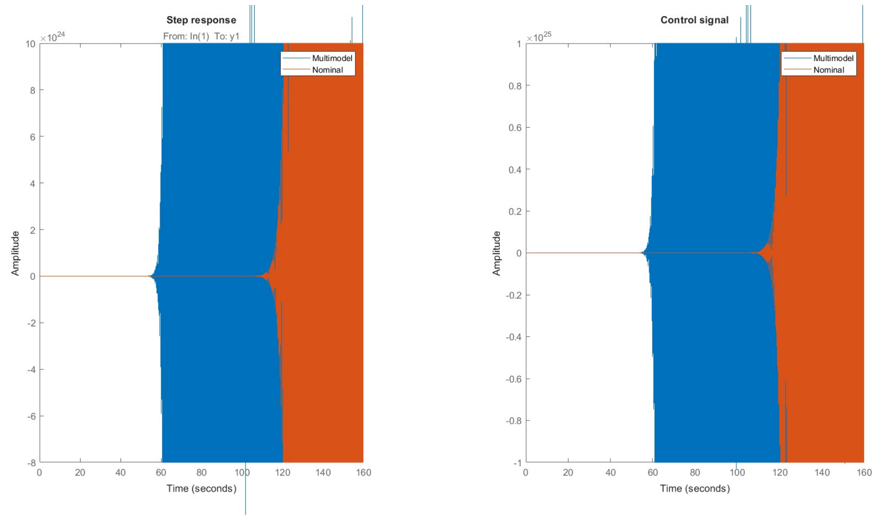


Figure 4.3: Step responses of closed-loop system and control signal - $\mathcal{H}_\infty$ order 5

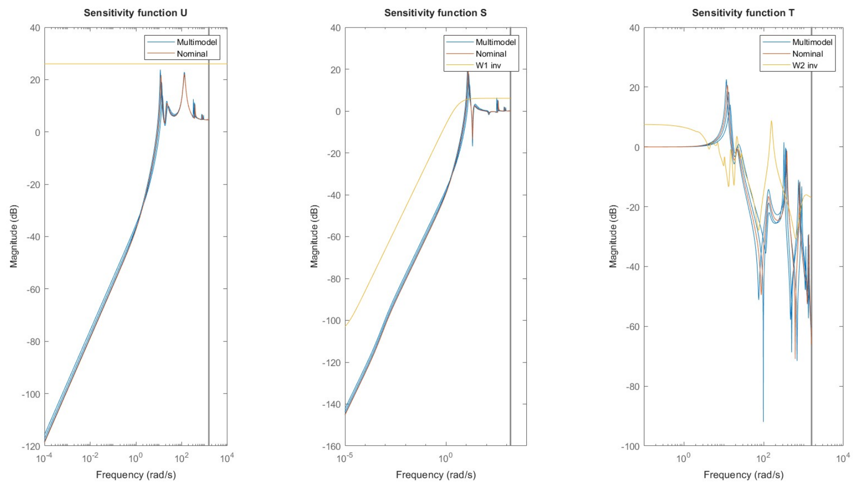


Figure 4.4: Bode of $\mathcal{U}$, $\mathcal{S}$, and $\mathcal{T}$ - $\mathcal{H}_\infty$ order 5

```
% 2. Comparison with model-based controller
load('H_inf_controller.mat')
Kinf = Kinf_3;

% Order of Kinf
orderK = size(Kinf.A, 1);
disp(['Order of the controller: ', num2str(orderK)])

% Reduction to order 5
Kreduced = reduce(Kinf, 5);
orderK = size(Kreduced.A, 1);
disp(['Order of the controller: ', num2str(orderK)])

Tnom = feedback(G_nom*Kreduced, 1);
Snom = feedback(1, G_nom*Kreduced);
Unom = feedback(Kreduced, G_nom);
T = feedback(Gmm*Kreduced, 1);
```

```
S = feedback(1, Gmm*Kreduced);
U = feedback(Kreduced, Gmm);

figure()
subplot(1,2,1)
step(T,Tnom)
title('Step response')
legend('Multimodel','Nominal')
subplot(1,2,2)
step(U,Unom)
title('Control signal')
legend('Multimodel','Nominal')

figure()
subplot(1,3,1)
bodemag(U,Unom,W3^-1)
title('Sensitivity function U')
legend('Multimodel','Nominal')
subplot(1,3,2)
bodemag(S,Snom, W1^-1)
title('Sensitivity function S')
legend('Multimodel','Nominal', 'W1 inv')
subplot(1,3,3)
bodemag(T,Tnom,W2^-1)
title('Sensitivity function T')
legend('Multimodel','Nominal','W2 inv')
```

4.1.3 Nominal performance and robust stability

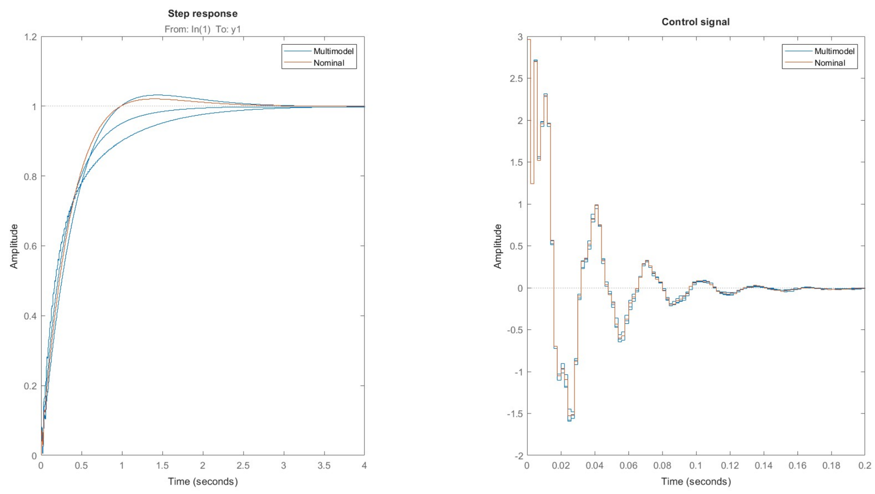


Figure 4.5: Step responses of closed-loop system and control signal

To achieve the shortest settling time, the robust stability constraint $\|W_2 T_{nom}\|_\infty < 1$ is enforced within the data-driven framework, while the optimization minimizes only the infinity norm:

$$\left\| \begin{bmatrix} W_1 S_{\text{nom}} \\ W_3 U_{\text{nom}} \end{bmatrix} \right\|_\infty$$

Results are presented in Figures 4.5 and 4.6. Figure 4.5 shows a shorter settling time, as evidenced by the step responses. Figure 4.6 confirms that nominal performance is achieved for all plant models, since the Bode magnitude of the sensitivity function $\mathcal{S}$ remains below $\mathcal{W}_1^{-1}$, consistent with Table 4.1. The infinity norm $\|W_2 T_{nom}\|_\infty$ is approximately 1, up to numerical error.

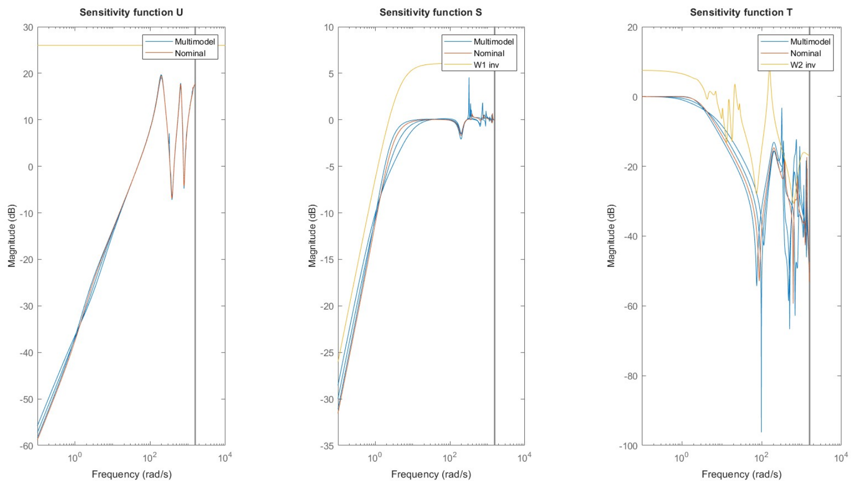


Figure 4.6: Bode of $\mathcal{U}$, $\mathcal{S}$, and $\mathcal{T}$

Table 4.1: $\|W_1 S\|_\infty$

G_i	Infinity Norm
G_1	0.6951
G_2	0.6259
G_3	0.6364
G_4	0.8411

```
% 3. Robust stability constraint
P_constraint = augw(G_nom, [], [], W2);
P_constraint = mktito(P_constraint, ny, nu);
P = augw(G_nom, W1, W3, []);
P = mktito(P, ny, nu);
K = datadriven.Controller.SS(order, ny, nu, Ts);
K.setinitial(K_c);
synthesiser = Synthesiser(K);
synthesiser.add_Hinf_objective(P, omegas);
synthesiser.add_Hinf_constraint(P_constraint, omegas);
synthesiser.ensure_controller_stability(omegas);

% Output
output = synthesiser.synthesise();
K_final_constrained = output.Controller;

% Plotting
Tnom = feedback(G_nom*K_final_constrained, 1);
Snom = feedback(1, G_nom*K_final_constrained);
Unom = feedback(K_final_constrained, G_nom);
```

```
T = feedback(Gmm*K_final_constrained, 1);
S = feedback(1, Gmm*K_final_constrained);
U = feedback(K_final_constrained, Gmm);

figure()
subplot(1,2,1)
step(T,Tnom)
title('Step response')
legend('Multimodel','Nominal')
subplot(1,2,2)
step(U,Unom)
title('Control signal')
legend('Multimodel','Nominal')

figure()
subplot(1,3,1)
bodemag(U,Unom,W3^-1)
title('Sensitivity function U')
legend('Multimodel','Nominal')
subplot(1,3,2)
bodemag(S,Snom, W1^-1)
title('Sensitivity function S')
legend('Multimodel','Nominal', 'W1 inv')
subplot(1,3,3)
bodemag(T,Tnom,W2^-1)
title('Sensitivity function T')
legend('Multimodel','Nominal','W2 inv')

nominal_performance = norm(W1*S, inf);
disp('Nominal performance')
disp(nominal_performance)

robust_stability = norm(W2*Tnom, inf);
disp('Robust stability')
disp(robust_stability)
```

4.2 Multimodel uncertainty

In this analysis, the data-driven approach models uncertainty as multimodel rather than multiplicative, while keeping the initial controller, its order, the weighting filter W_3 (and W_1 in the baseline case), and the frequency grid unchanged. Since the optimization is carried out simultaneously on all four models, the filter W_2 is no longer required.

4.2.1 Data-driven control

Analogously to the previous case, the LFT method is used to define the augmented plant, but now for all four systems. Consequently, the optimization problem is to minimize:

$$\left\| \begin{bmatrix} W_1 S_i \\ W_3 U_i \end{bmatrix} \right\|_\infty \qquad \forall\, i = 1, \dots, 4.$$

Figure 4.7 shows the step responses, which exhibit a shorter settling time than those in Section 4.1. Figure 4.8 displays the Bode diagrams of the sensitivity functions for the multimodel design.

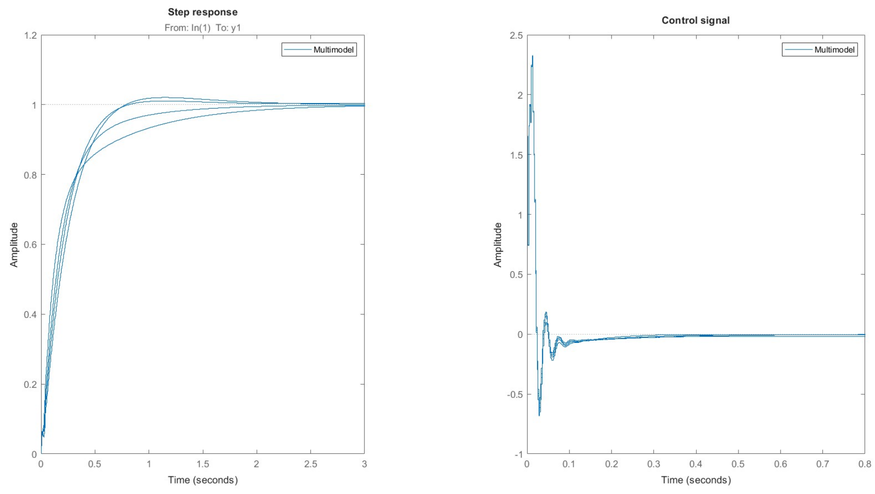


Figure 4.7: Step responses of closed-loop system and control signal - multimodel

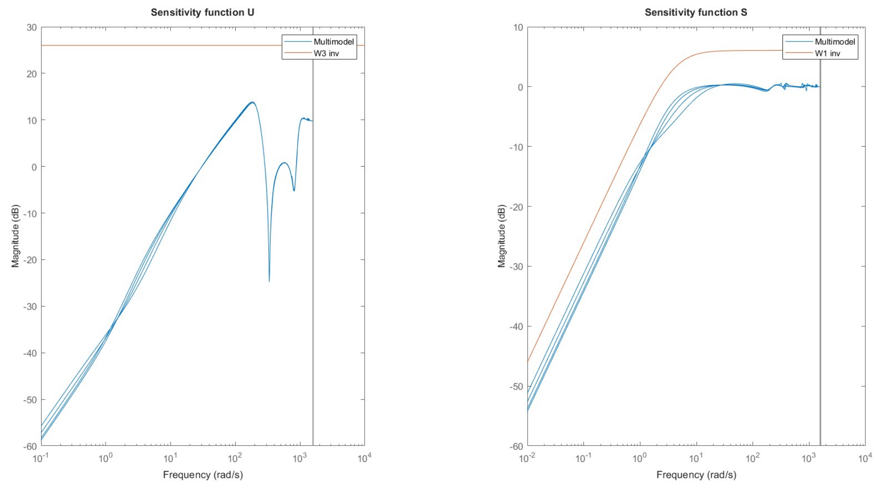


Figure 4.8: Bode of $\mathcal{U}$ and $\mathcal{S}$ - multimodel

```
% 1. Proportional initial controller
order = 5;
ny = 1;
nu = 1;
K_c = c2d(tf(1), Ts, 'zoh');

% Check that K_c is a stabilizing controller
Gmm = stack(1, G_struct(1).G, G_struct(2).G, ...
    G_struct(3).G, G_struct(4).G);
isstable(feedback(1, Gmm))

% LFR model
P = augw(Gmm, W1, W3, []);
P = mktito(P, ny, nu);

% Problem formulation
K = datadriven.Controller.SS(order, ny, nu, Ts);
K.setinitial(K_c);

% Objective and ensuring stability (not needed in bounded real lemma)
synthesiser = Synthesiser(K);
synthesiser.add_Hinf_objective(P, omegas);
synthesiser.ensure_controller_stability(omegas);

% Output
output = synthesiser.synthesise();
K_final = output.Controller;

% Plotting
T = feedback(Gmm*K_final, 1);
S = feedback(1, Gmm*K_final);
U = feedback(K_final, Gmm);

figure()
subplot(1,2,1)
step(T)
title('Step response')
legend('Multimodel')
subplot(1,2,2)
step(U)
title('Control signal')
legend('Multimodel')

figure()
subplot(1,2,1)
bodemag(U,W3^-1)
title('Sensitivity function U')
legend('Multimodel','W3 inv')
subplot(1,2,2)
bodemag(S,W1^-1)
title('Sensitivity function S')
legend('Multimodel','W1 inv')
```

4.2.2 Benefits on the desired settling time

To highlight the impact of a multimodel approach on settling time, a more aggressive weighting filter is adopted by shifting the zero of the continuous-time equivalent of W_1 from -4 to -20:

$$W_1(z) = \frac{0.5\,z - 0.48}{z - 1}.$$

Controllers are then synthesized for both the multiplicative and the multimodel uncertainty cases; the corresponding results are shown in Figures 4.9 and 4.10. A comparison of the step responses reveals that the signal settles much faster in the multimodel case, demonstrating that this approach achieves a lower settling time. The improvement arises because the data-driven design directly incorporates all four plant models, rather than representing their discrepancies by the uncertainty filter W_2; the optimization is thus guided simultaneously by the behavior of all models.

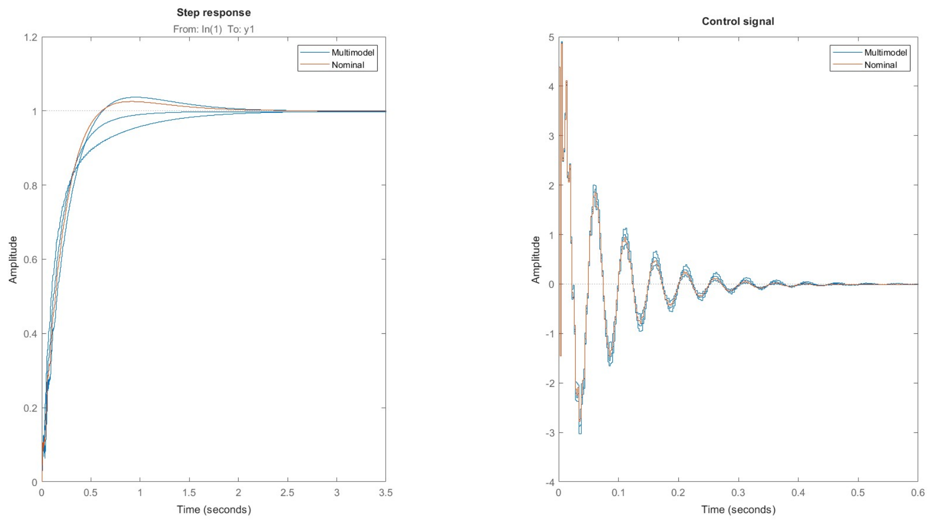


Figure 4.9: Step responses of closed-loop system and control signal - faster multiplicative

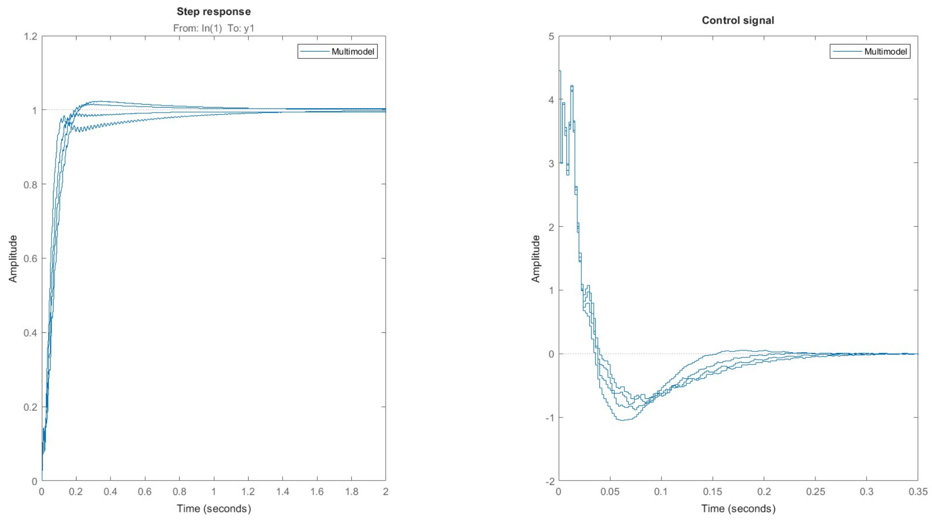


Figure 4.10: Bode of $\mathcal{U}$ and $\mathcal{S}$ - faster multiplicative

```
% 2. Improved settling time (comparison)
num = [1, 20];
den = [1, 1e-5];
W1 = tf(num, den)*1/2;
W1 = c2d(W1, Ts, 'zoh');

% Multiplicative
P = augw(G_nom, W1, W3, W2);
P = mktito(P, ny, nu);
K = datadriven.Controller.SS(order, ny, nu, Ts);
K.setinitial(K_c);
synthesiser = Synthesiser(K);
synthesiser.add_Hinf_objective(P, omegas);
synthesiser.ensure_controller_stability(omegas);
output = synthesiser.synthesise();
K_final = output.Controller;

% Plotting
Tnom = feedback(G_nom*K_final, 1);
Snom = feedback(1, G_nom*K_final);
Unom = feedback(K_final, G_nom);
T = feedback(Gmm*K_final, 1);
S = feedback(1, Gmm*K_final);
U = feedback(K_final, Gmm);

figure()
subplot(1,2,1)
step(T,Tnom)
title('Step response')
legend('Multimodel','Nominal')
subplot(1,2,2)
step(U,Unom)
title('Control signal')
legend('Multimodel','Nominal')

figure()
subplot(1,3,1)
bodemag(U,Unom,W3^-1)
title('Sensitivity function U')
legend('Multimodel','Nominal')
subplot(1,3,2)
bodemag(S,Snom, W1^-1)
title('Sensitivity function S')
legend('Multimodel','Nominal', 'W1 inv')
subplot(1,3,3)
bodemag(T,Tnom,W2^-1)
title('Sensitivity function T')
legend('Multimodel','Nominal','W2 inv')

% Multimodel
P = augw(Gmm, W1, W3, []);
P = mktito(P, ny, nu);
K = datadriven.Controller.SS(order, ny, nu, Ts);
K.setinitial(K_c);
synthesiser = Synthesiser(K);
```

```
synthesiser.add_Hinf_objective(P, omegas);
synthesiser.ensure_controller_stability(omegas);
output = synthesiser.synthesise();
K_final = output.Controller;

% Plotting
T = feedback(Gmm*K_final, 1);
S = feedback(1, Gmm*K_final);
U = feedback(K_final, Gmm);

figure()
subplot(1,2,1)
step(T)
title('Step response')
legend('Multimodel')
subplot(1,2,2)
step(U)
title('Control signal')
legend('Multimodel')

figure()
subplot(1,2,1)
bodemag(U,W3^-1)
title('Sensitivity function U')
legend('Multimodel','W3 inv')
subplot(1,2,2)
bodemag(S,W1^-1)
title('Sensitivity function S')
legend('Multimodel','W1 inv')
```

4.2.3 Advantages and drawbacks of the data-driven approach

The data-driven approach offers several advantages over the model-based one. First, it does not require a parametric model - unlike the approximate identification in Chapter 1 - and relies solely on frequency-response data. Second, it produces fixed-structure, low-order controllers, a feature that the model-based framework cannot guarantee (see Section 4.1). Third, it can directly handle multimodel uncertainty, as demonstrated in Subsection 4.2.1, leading to significantly shorter settling times. Other advantages, not explored here, include the implicit handling of time delays.

The method has some drawbacks. It requires an initial stabilizing controller: in this case, the nominal plant G_{nom} was unstable, and $K_c = 1$ stabilized it - yet choosing a much smaller K_c could have prevented convergence. Moreover, the frequency range used for optimization must be carefully selected to ensure stable optimization.