ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

CE2: ROBUST CONTROL OF AN ELECTRO-MECHANICAL SYSTEM

ADVANCED CONTROL SYSTEMS ME-524

PROF. ALIREZA KARIMI

GROUP B

Angelo Giovine 368440

Baptiste Bühler 326168

14th August 2024

Contents

CHAPTER 1

MULTIPLICATIVE UNCERTAINTY

In this section, we are going to study an electro-mechanical system, a Quanser Servo-Qube (Figure 1.1) with weights attached on top. Attaching varied weights to different locations on the flexible component can lead to diverse loadings, thereby introducing multimodal uncertainty. Our aim is to translate this uncertainty into a multiplicative one. 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 $\|\Delta\|_\infty \leq 1$ and $G(s)$ is the nominal model of $\tilde{G}(s)$.

FIGURE 1.1
Quanser Servo-Qube

1.1

Firstly given a set of measurements, for different loads, we are going to obtain the parametric model and non parametric model. For the parametric model we are using the MATLAB `oe` function, that generates a conventional transfer functions that relate measured inputs to outputs while also including white noise as an additive output disturbance in the form:

$$y(t) = \frac{B(q)}{F(q)}u(t - nk) + e(k)$$

We choose 8 as order of the polynomial $B(q)$ and $F(q)$. For the non-parametric model we use the `spa` model. Here the relative code :

LISTING 1.1
Parametric and non-parametric model

```
load logs_1.mat
G1 = oe(data, [8 8 1]); % Parametric
Ts = G1.Ts; % Same for all data
freqs = (pi/4096:pi/4096:pi) / Ts; % Same for all data
Gf1 = spa(data,8191,freqs); % Non-parametric
```

1.2

Now we are interested in the Nyquist plot of both versions of the identified system, with their uncertainties with a 95 % confidence level:

LISTING 1.2
Nyquist

```
opts = nyquistoptions;
opts.ConfidenceRegionDisplaySpacing = 3;
opts.ShowFullContour = 'off';

figure(1)
nyquistplot(Gf1,G1,freqs,opts,'sd',2.45);
legend('Non-parametric','Parametric')
```

1.3

From the data presented in Figure 1.2, it is observed that the confidence regions exhibit varied geometries for the different models:

- For the parametric model, the uncertainty areas are elliptical, reflecting the multivariate Gaussian nature of the uncertainty, with a diagonal covariance matrix
- Conversely, the non-parametric model's uncertainty areas are circular, a consequence of the real and imaginary components of the transfer function, $\hat{G}(e^{j\omega})$, being asymptotically independent and normally distributed with equal variance $\frac{\Phi_V(\omega)}{2\Phi_u(\omega)}$.

Furthermore, the larger size of the uncertainty bounds for the non-parametric model as compared to the parametric model suggests a reduced precision in the estimates derived from the non-parametric approach.

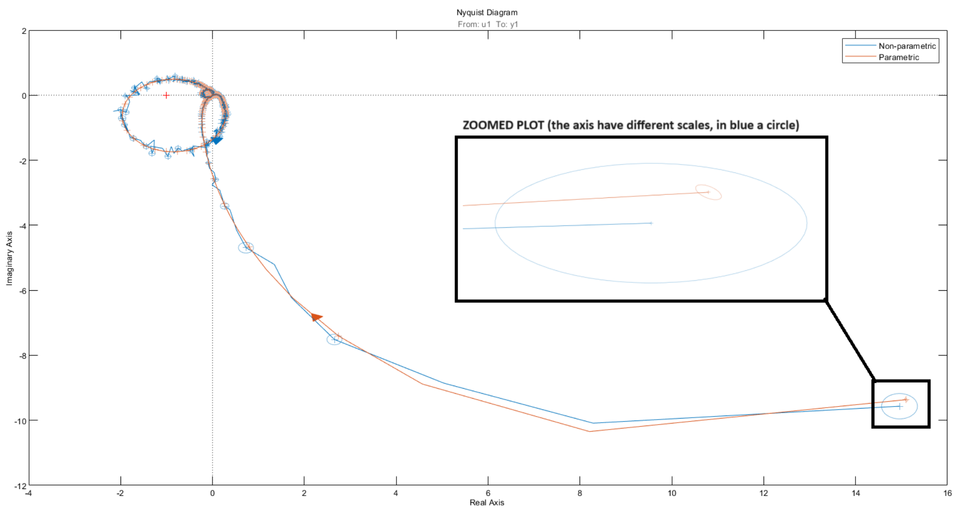

FIGURE 1.2
Nyquist of parametric and non-parametric model

1.4

In the same way of Section 1.1, we are going to build both types of models for all the available data, at the end of the process our uncertain model is represented by $\mathcal{G} = \{G_1, G_3, , G_5, G_7, G_9, G_{11}\}$

LISTING 1.3
Multi

```
Gmm = stack(1,G1,G3,G5,G7,G9,G11);
```

1.5

By choosing G_1 as the nominal model we can compute the weighting filter W_2 and plotting it by running the following code:

LISTING 1.4
W2 with G1 as nominal system

```
[Gu1, info1] = ucover(Gmm, G1, 2);
W2_1 = info1.W1opt;
figure(8) % Gnom = G1
hold on
bodemag((G3/G1)-1, 'b')
bodemag((G5/G1)-1,'b')
bodemag((G7/G1)-1,'b')
bodemag((G9/G1)-1,'b')
bodemag((G11/G1)-1,'b')
bodemag(W2_1,'r')
title('Nominal model : G1')
legend('G3/G1)-1','(G5/G1)-1','G7/G1)-1','(G9/G1)-1','(G11/G1)-1','W2_1')
```

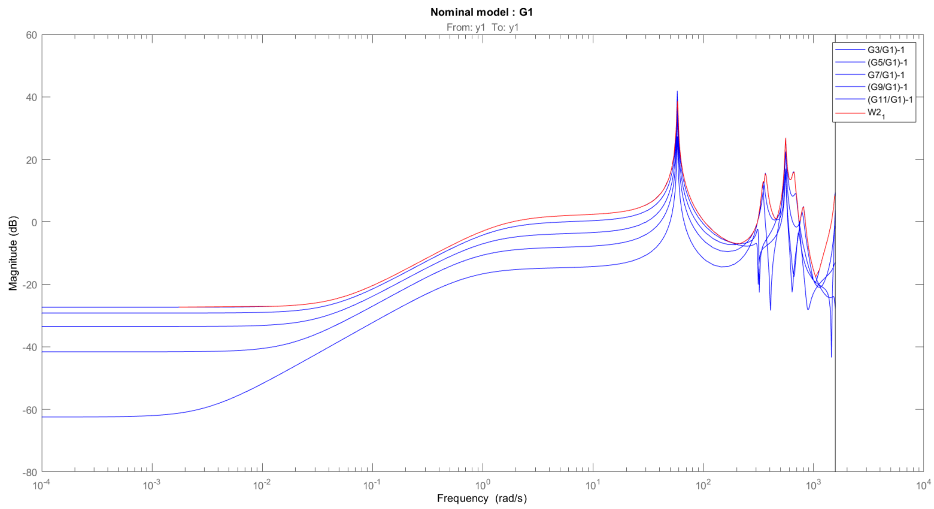

FIGURE 1.3
W_2 when G_1 is chosen as nominal system

You can see on the Fig. 1.4 that the weighting filter W_2 is cutting the peak of the propagation error. This is due to a sampling error. The peak frequency is not sampled and thus, the filter does not get the information of the maximal peak value. To avoid this, it is possible to use the non-parametric model, but the output is a noisy weighting filter when computing the optimal one.

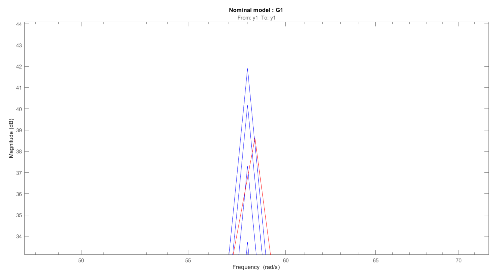

FIGURE 1.4
W_2 is cutting the peak

1.6

To select the optimal nominal model, our strategy entails graphically representing the filters derived from each system when treated as the nominal system and conducting comparisons among them. As nominal model we will choose the one that will produce the smallest $|W_2(j\omega)|$.

To do that we observe Figure 1.5 and we notice that for high frequencies (bigger than $10^3 rad/s$) all the filters are comparable, while at low frequencies it's evident the lower magnitude of the filter obtained when G_7 is chosen as nominal model. Hence it's reasonable to choose $G_{nom} = G_7$.

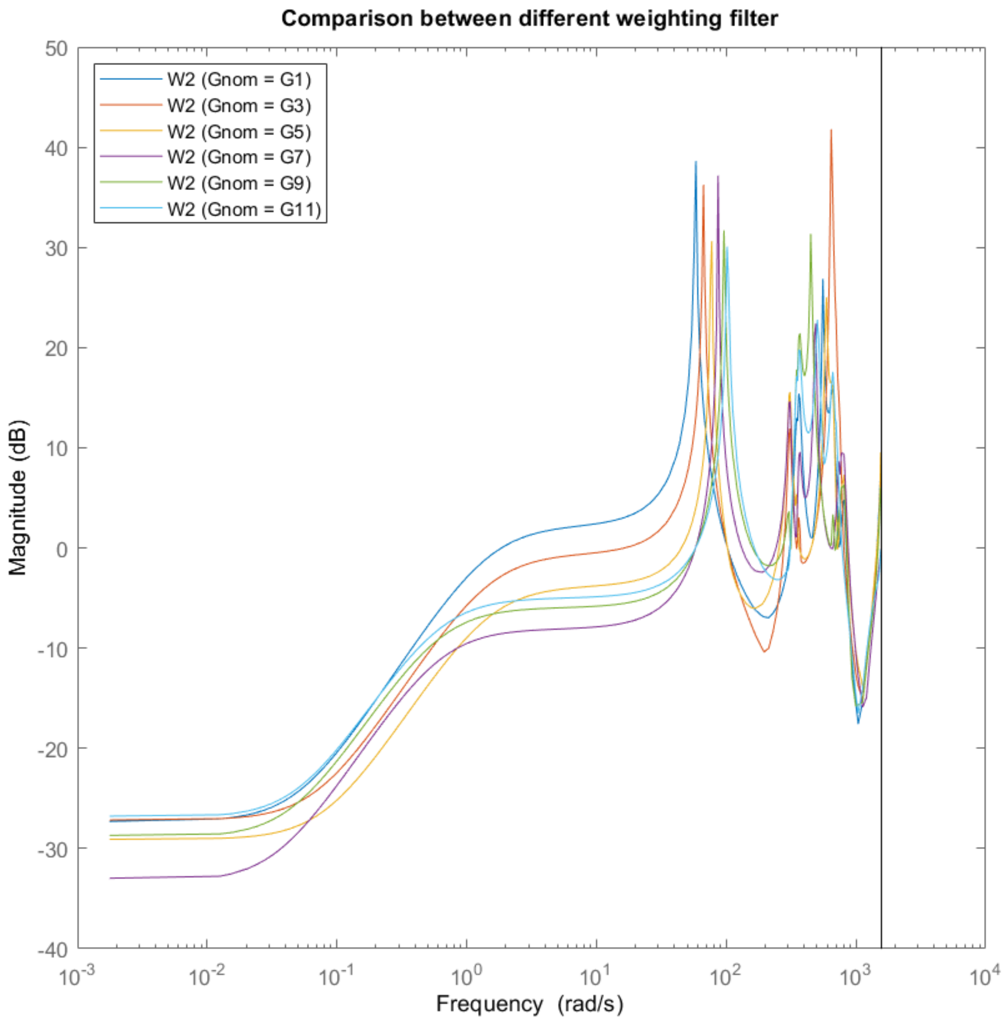

FIGURE 1.5
Comparison between different W2

Chapter 2

Model-Based $\mathcal{H}_\infty$ Controller Design

Here we want to design a state feedback controller such that the ∞-norm of the closed-loop transfer functions from the input disturbance to the output and to the input of the system are minimized. In this section we use as nominal model the model G_7.

2.1

First we design a weighting filter $W_1(z)$ regarding these proprieties :

1. Zero steady-state tracking error for a step reference, hence we want an integral action in the controller. To achieve that, our filter need an integrator, to avoid numerical problems we have used a quasi-integrator.
2. A modulus margin of at least 0.5, so we need the HF gain for W_1^{-1} to be 0.5
3. The shortest settling time for the nominal model

We choose a weighting filter $W_1(z)$:

$$W_1(z) = \frac{0.5z - 0.493}{z - 1} \tag{2.1}$$

The settling time using this filter is 0.49s

Listing 2.1
Design of W1

```
num = [1 7];
den = [1 0.0001];
W1 = tf(num,den) * 1/2;
W1 = c2d(W1,Ts,'zoh');
```

In Fig. 2.1 we can observe that as required the magnitude of W_1^{-1} is less than 6dB.

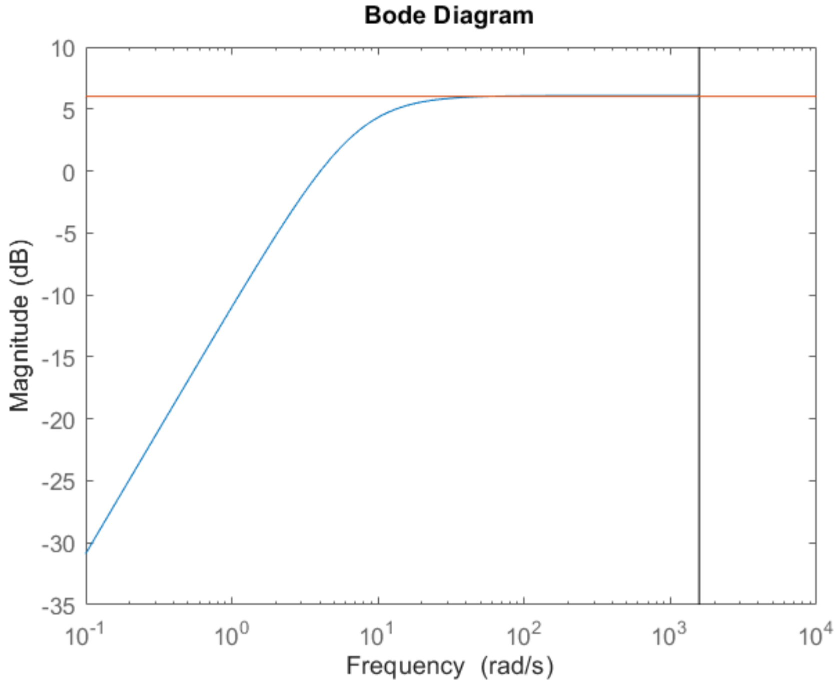

FIGURE 2.1
Bode plot of the magnitude of the inverse of W_1

2.2

In this section we want to find the controller K_{H_∞} for robust performance. We use the mixed sensitivity approach with the MATLAB function `mixsyn`. This function provide a controller that minimize the H_∞ norm of the weighted closed-loop transfer function :

$$M(s) = \begin{bmatrix} W_1\mathcal{S} \\ W_2\mathcal{U} \\ W_3\mathcal{T} \end{bmatrix}$$

where we defined :

$$\mathcal{S} = \frac{1}{1+GK} \qquad \mathcal{U} = K\mathcal{S} = \frac{K}{1+GK} \qquad \mathcal{T} = \frac{GK}{1+GK}$$

And this specific application we have as performance filter the filter W_1 defined in equation 2.1. The weighting filter on the input sensitivity function, named W_2 in `mixsyn` function, is zero here. The weighting filter on the function $\mathcal{T}$, named W_3, is the one obtained using the function `ucover` for the nominal model G_7 with an order 4. We named it W_2.

LISTING 2.2
Definition of W2

```
[Gu74, info74] = ucover(Gmm, G7, 4);
W2 = info74.W1;
Gnom = G7;
```

Then, we obtain the controller in a state-space model.

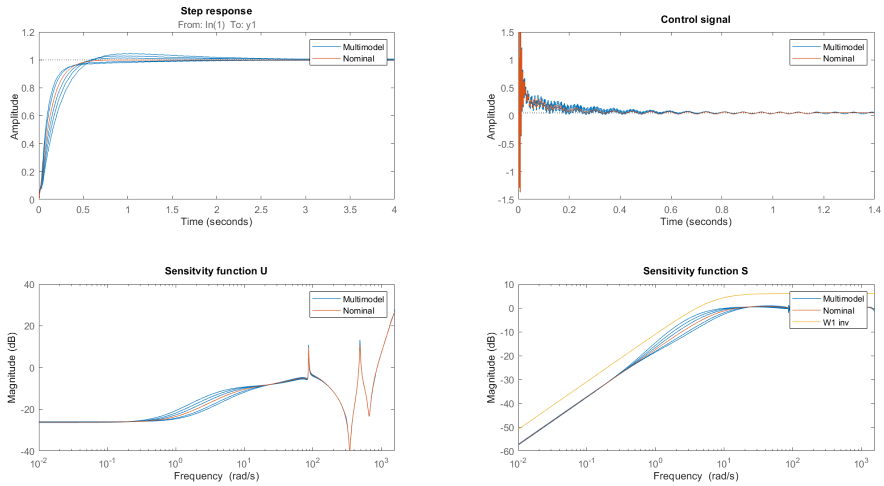

FIGURE 2.2
Step response of the closed-loop system, step response of the control signal, Bode of the sensitivity functions $\mathcal{U}$ and $\mathcal{S}$

LISTING 2.3
H∞ controller

```
Kinf = mixsyn(Gnom,W1,[],W2);
```

2.3

In this section we want to plot the step response of the closed-loop system, so the output and the control signal. We want also the magnitude of the input sensitivity function $\mathcal{U}(z)$ and the sensitivity function $\mathcal{S}(z)$. To find the transfer functions $\mathcal{S},\mathcal{T},\mathcal{U}$ we use the MATLAB function `feedback`.

LISTING 2.4
Definition of $\mathcal{S},\mathcal{T},\mathcal{U}$

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

We obtain the results shown in Fig. 2.2. We can see that the performance condition is satisfied. But we can also see that the sensitivity function $\mathcal{U}$ has very high magnitude at high frequencies, which leads to large values at the beginning of the step response of the control signal.

Furthermore, the robust performance conditions are met because :

$$\left\| \begin{bmatrix} W_1\mathcal{S} \\ W_2\mathcal{T} \end{bmatrix} \right\|_\infty = 0.6076 < \frac{1}{\sqrt{2}}$$

LISTING 2.5
Robust performance conditions check

```
condition = norm([W1*Snom W2*Tnom],inf)
if (condition <= 1/sqrt(2))
    fprintf('Robust performance conditions are met\n')
end
```

2.4

Related to this large magnitude of the control signal when a unit step reference is applied, we want our control signal $u(t)$ to be within the range ± 1.5V. To reduce the magnitude of the control signal, it is possible to apply a third filter, named W_2 in `mixsyn`. From the definition of `mixsyn`, we can also include the sensitivity function $\mathcal{U}$ in the calculations. To do so, we choose a filter $W_3 = 0.1$ by trial and error. Then, we obtain new results shown in Fig. 2.3.

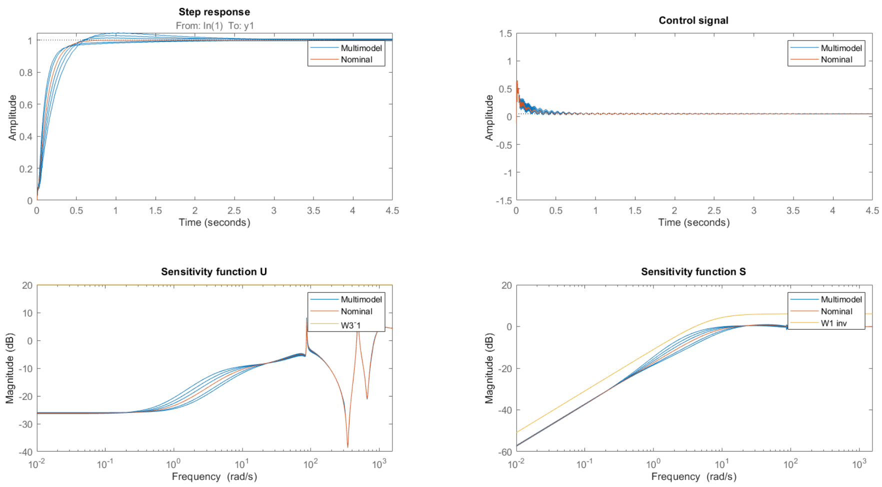

FIGURE 2.3
Step response of the closed-loop system, step response of the control signal, Bode of the sensitivity functions $\mathcal{U}$ and $\mathcal{S}$ with a range for the control signal of ± 1.5V

We can observe on the Fig. 2.3 that the sensitivity function $\mathcal{U}$ is reduced at high frequencies, thus the control signal is also reduced and we obtain the required range. Furthermore the robust performance conditions are met because :

$$\left\| \begin{bmatrix} W_1\mathcal{S} \\ W_2\mathcal{T} \end{bmatrix} \right\|_\infty = 0.6087 < \frac{1}{\sqrt{2}}$$

LISTING 2.6
Definition of W3 and redefinition of the controller

```
W3 = 0.1;
Kinf_range = mixsyn(Gnom, W1, W3, W2);

T = feedback(Gmm*Kinf_range,1);
Tnom = feedback(Gnom*Kinf_range,1);
U =feedback(Kinf_range,Gmm);
Unom =feedback(Kinf_range,Gnom);
S = feedback(1,Gmm*Kinf_range);
```

```
Snom = feedback(1,Gnom*Kinf_range);
```

2.5

In this section we want to possibly reduce the order of our controller K_{H_∞}. The actual order of the controller is 13. After looking at its pole-zero map (Fig. 2.4, using `pzmap`. We can observe some poles and zeros that are really close

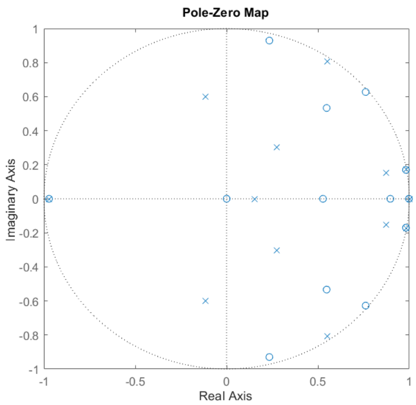

FIGURE 2.4
Pole-zero map of the controller before reduction of the order

We use the function `reduce` to get a controller of 11th-order. We choose this order to obtain a controller satisfying the robust performance conditions. We can compare the two controllers in Fig. 2.5 and see that they are very similar even though there is a difference of two orders.

LISTING 2.7
Reduction of the order of the controller

```
K_ft = ss2tf(Kinf_range.A,Kinf_range.B,Kinf_range.C,Kinf_range.D);
Kreduced = reduce(Kinf_range, 11);
```

Then we obtain the response with the reduced controller in Fig. 2.6 and see that the robust performance stability are met because :

$$\left\| \begin{bmatrix} W_1\mathcal{S} \\ W_2\mathcal{T} \end{bmatrix} \right\|_\infty = 0.6756 < \frac{1}{\sqrt{2}}$$

2.6

In this section, we want to discuss the closed-loop norm :

$$\left\| \begin{bmatrix} W_1\mathcal{S} \\ W_2\mathcal{T} \end{bmatrix} \right\|_\infty$$

when using the nominal model G_{nom} or the multimodel set $_{Gmm}$.

We obtain these results :

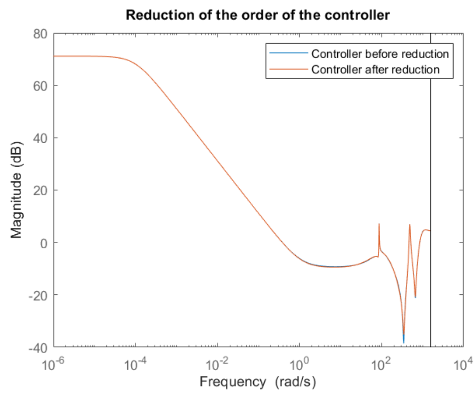

FIGURE 2.5
Comparison between the initial controller and the reduced one

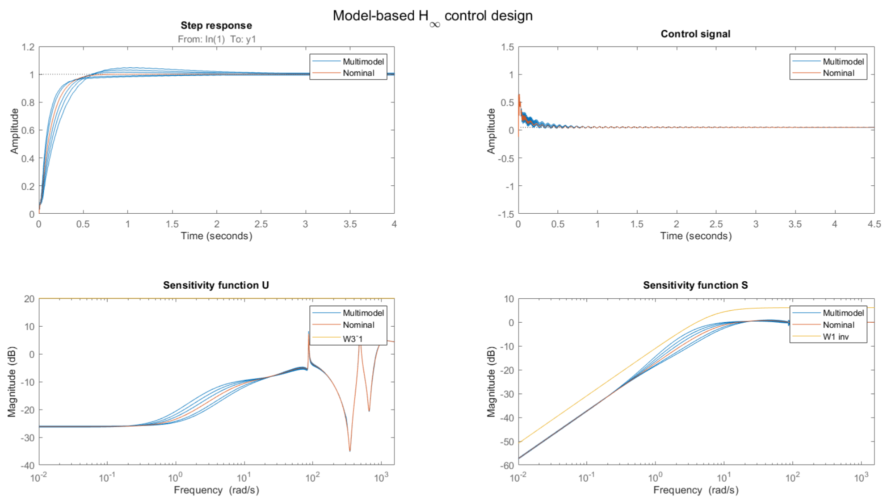

FIGURE 2.6
Step response of the closed-loop system, step response of the control signal, Bode of the sensitivity functions $\mathcal{U}$ and $\mathcal{S}$ with a range for the control signal of ± 1.5V. The controller is reduced to the 11th-order

$$\text{norm nominal} = 0.6756 \qquad \text{norm multimodel} = \begin{bmatrix} 15.9244 \\ 11.0955 \\ 5.6489 \\ 0.6756 \\ 6.2901 \\ 11.1381 \end{bmatrix}$$

We can see that the norm of the multimodel contains the norm of the nominal model. We can also see that the norm of the nominal model is the smallest norm in the multimodel and also the only one less than 1, this is due to the aim of the computation of the $\mathcal{H}_\infty$.

LISTING 2.8
Discussion on the ∞-norm

```
norm_nominal = norm([W1*Snom W2*Tnom],inf)
norm_multimodel = norm([W1*S W2*T],inf)
```

CHAPTER 3

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

Here we want to design a state feedback controller such that the sum of the (squared) two-norm of the closed-loop transfer functions from the input disturbance to the output and to the input of the system are minimized.

3.1

First it is necessary to convert our system from discrete time to continuous time to then derive a controller.

LISTING 3.1
from dt to ct

```
% Conversion from discrete time to continuous time
Gct = d2c(Gnom);
[A,B,C,D] = ssdata(Gct);
```

3.2

In the case of a strictly proper system the state-space equations are:

$$\begin{cases} \dot{x}(t) = Ax(t) + Bu(t) \\ y(t) = Cx(t) \end{cases} \tag{3.1}$$

In the case of a closed-loop with a state feedback controller, we can write the input $u(t)$ as:

$$u(t) = -Kx(t) + v(t) \tag{3.2}$$

where $v(t)$ is the input disturbance.
In order to minimize the two transfer functions we define $y_1(t) = Cx(t)$ and $y_2(t) = -Kx(t)$ and the resulting state-space equation is:

$$\begin{cases} \dot{x}(t) = (A - BK)x(t) + Bv(t) \\ y_1(t) = Cx(t) \\ y_2(t) = -Kx(t) \end{cases} \tag{3.3}$$

Hence in our case we want to minimize:

$$||G_{v \to y_1}||_2^2 + ||G_{v \to y_2}||_2^2 \tag{3.4}$$

3.3

Using the Bounded $\mathcal{H}_2$ theorem we can state the problem as an optimization one:

$$\begin{aligned} \min \ & \text{trace}(CLC^T) + \text{trace}(KLK^T) \\ \text{s.t.} \ & (A - BK)L + L(A - BK)^T + BB^T \preceq 0 \\ & L \succ 0 \end{aligned}$$

But since in both the objective function and in the constraints we have multiplication between decision variables, we should restate our formulation to have LMIs objective and constraints.

For the constraint a simple trick is to define a new variable $X = KL$, hence the first constraint would become:

$$AL + LA^T - BX - X^T B^T + BB^T \preceq 0 \tag{3.5}$$

Now X appears linearly, so the stated equation is an LMI.

For the objective function, we can define a new variable $M \succeq KLK^T$, but we should translate this new induced constraint into an LMI. To do that first we substitute KL with X and K^T with LX^{-1}, obtaining:

$$M - XLX^{-1} \succeq 0 \tag{3.6}$$

Then by applying the Schur theorem, we can translate this equation into an LMI obtaining:

$$\begin{bmatrix} M & X \\ X^T & L \end{bmatrix} \succeq 0 \tag{3.7}$$

In the end, the finial formulation for the convex optimization problem becomes:

$$\begin{aligned} \min \ & \text{trace}(CLC^T) + \text{trace}(M) \\ \text{s.t.} \ & AL + LA^T - BX - X^T B^T + BB^T \preceq 0 \\ & \begin{bmatrix} M & X \\ X^T & L \end{bmatrix} \succeq 0 \\ & L \succ 0 \end{aligned} \tag{3.8}$$

Finally we can compute the controller by solving this optimization problem using MOSEK, here is the relative code:

LISTING 3.2
2-Norm controller

```
% Optimization problem definition:
n = size(A,1);
```

```
m = size(B,2);
% decision variables:
L = sdpvar(n,n,'symmetric');
X = sdpvar(m,n);
M = sdpvar(m,m);
%objective function:
obj = trace(C*L*C') + trace(M);
% lmis definition:
lmi1 = A*L -B*X + L*A' - X'*B' + B*B' <= 0;
lmi2 = [M X; X' L] >= 0;
lmi3 = L >= 0;
lmi = [lmi1,lmi2,lmi3];
% options:
options = sdpsettings('solver','mosek');
optimize(lmi,obj,options);
% controller:
X = value(X);
L = value(L);
K_H2 = X*inv(L);
```

3.4

The resulting controller is:

$$K_{H_2} = \begin{bmatrix} 2.0000 & 0.4656 & 2.8658 & 1.2251 & 2.4430 & 1.7477 & 1.4860 & 2.6511 & 9.8458 \end{bmatrix} \tag{3.9}$$

3.5

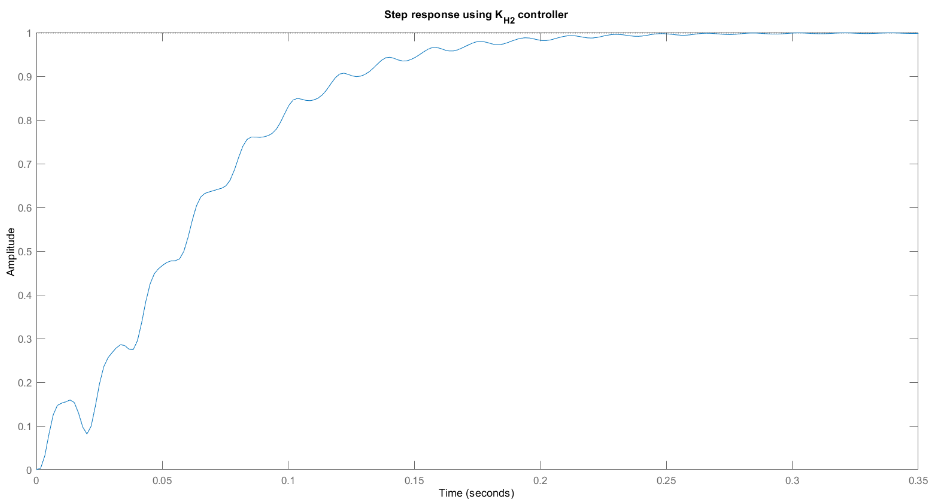

FIGURE 3.1
Step response of the closed-loop system using K_{H_2}

3.6

For a continuous time system, LQR computes the state feedback control $u = -Kx$ that minimizes the quadratic cost function:

$$J(u) = \int_0^{\infty} \left(x^T Q x + u^T R u\right) dt \tag{3.10}$$

If we consider $Q = C^T C$ and $R = 1$ we have that:

$$J(u) = \int_0^{\infty} |y|^2 + |u|^2 dt = \int_0^{\infty} |y_1|^2 + |y_2|^2 dt \tag{3.11}$$

Since we know that both $y_1(t)$ and $y_2(t)$ are equal to zero for $t < 0$, if we apply Parseval's theorem, we have that minimizing (3.11) is the same as minimizing:

$$J(u) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} |Y_1(s)|^2 ds + \frac{1}{2\pi} \int_{-\infty}^{+\infty} |Y_2(s)|^2 ds \tag{3.12}$$

which can be rewritten as:

$$J(u) = ||Y_1(s)||_2^2 + ||Y_2(s)||_2^2 \tag{3.13}$$

and since $V(s)$ is fixed, minimizing $J(u)$ is the same as minimizing:

$$||G_{v \to y_1}||_2^2 + ||G_{v \to y_2}||_2^2 \tag{3.14}$$

For this reason, we expect the LQR controller to be the same as the $\mathcal{H}_2$ controller.

The MATLAB code to compute the LQR controller is:

LISTING 3.3
LQR

```
% LQR
Q = C'*C;
R = eye(m);
[K_lqr,~,~] = lqr(A,B,Q,R);
```

The resulting controller is:

$$K_{LQR} = \begin{bmatrix} 2.0000 & 0.4656 & 2.8658 & 1.2251 & 2.4430 & 1.7477 & 1.4860 & 2.6511 & 9.8458 \end{bmatrix} \tag{3.15}$$

As expected the controllers K_{LQR} and K_{H_2} are the same, hence the step response is the same (Figure 3.2).

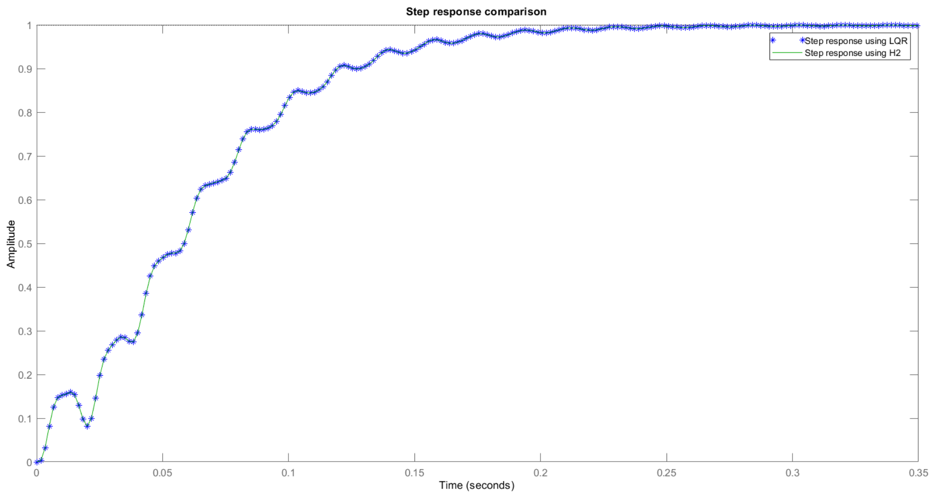

FIGURE 3.2
Comparison of the two closed-loop systems step response

Chapter 4

Data-driven controller Multiplicative uncertainty

Here we want to compute a $\mathcal{H}_\infty$ controller by using the frequency-response of our system.
We will use the provided `datadriven.m` function to implement the data-driven approach.

Firstly we define the parameters that we are going to use for the controller identification:

- *The initial stabilizing controller*: We are taking as initial controller K_c a small gain integrator controller in the form: $K_c = \frac{c}{1-z^{-1}}$, where we have chosen $c = 0.001$, and verified that the resulting controller was stabilizing.
- *F_x and F_y* : Since $K = XY^{-1}$, observing our initial controller K_c we will choose as fixed part of the X, $F_x = z$ and as fixed part of Y, $F_y = z - 1$.
- *The required order of the final controller*: we will choose 11
- *The frequency grid*: we have defined a frequency grid composed of 400 logarithmically spaced frequencies between 0 and π/T_s.

The minimization problem we want to solve has objective function:

$$\min \left\| \begin{bmatrix} W_1\mathcal{S} \\ W_2\mathcal{T} \\ W_3\mathcal{U} \end{bmatrix} \right\|_\infty \tag{4.1}$$

where $W_1(s)$ is the filter designed in Section 2.1,$W_2(s)$ is the filter designed in Section 1.6, while $W_3(s)$ is the filter designed in Section 2.4

4.1

The resulting controller is:

```
K =

   0.8729 z^11 - 2.043 z^10 + 1.895 z^9 - 0.5845 z^8 - 0.5 z^7 + 0.4555 z^6 + 0.1278 z^5 - 0.3003 z^4 - 0.2262 z^3 + 0.435 z^2 - 0.1904 z + 0.0593
  ---------------------------------------------------------------------------------------------------------------------------------------------------
  z^11 - 0.7072 z^10 - 0.4374 z^9 + 0.03228 z^8 + 0.02915 z^7 - 0.1737 z^6 - 0.07207 z^5 + 0.2271 z^4 + 0.1325 z^3 + 0.1617 z^2 - 0.05927 z - 0.133
```

4.2

To assert our performance and our robustness we can do the same check of chapter 2

$$\left\| \begin{bmatrix} W_1 \mathcal{S} \\ W_2 \mathcal{T} \end{bmatrix} \right\|_\infty = 0.6130 < \frac{1}{\sqrt{2}} \tag{4.2}$$

Both the conditions are met, moreover, Figure 4.2, helps us to validate our design.

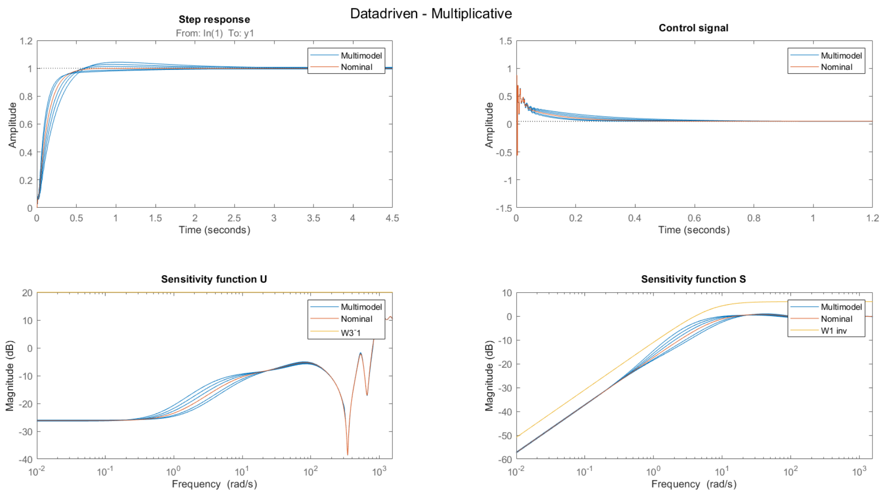

FIGURE 4.1
Data driven Multiplicative uncertainty

To compare this design, with the other approach of Chapter 2, we can observe Figure 4.3. As expected, the system behavior is almost the same. In both cases, if we look at the step response, the nominal system is the one with less overshoot and the smallest settling time. The only difference is in the control signal, and in the sensitivity function $\mathcal{U}$, but in both cases the constraints are satisfied.

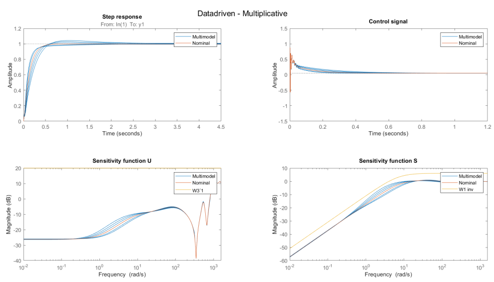

FIGURE 4.2
Data driven Multiplicative uncertainty

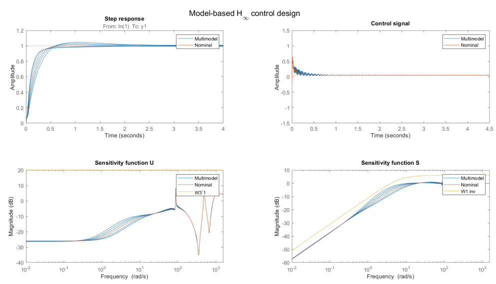

FIGURE 4.3
Mixed sensitivity Matlab

4.3

In this section, we will develop a reduced-order controller.

Using a trial and error method, we successfully decreased the order while ensuring all specifications were met. The final order was reduced to 8. As shown in Figure 4.4, this reduction in order results in a slight overshoot in the step response, and adversely impacts the control signal.

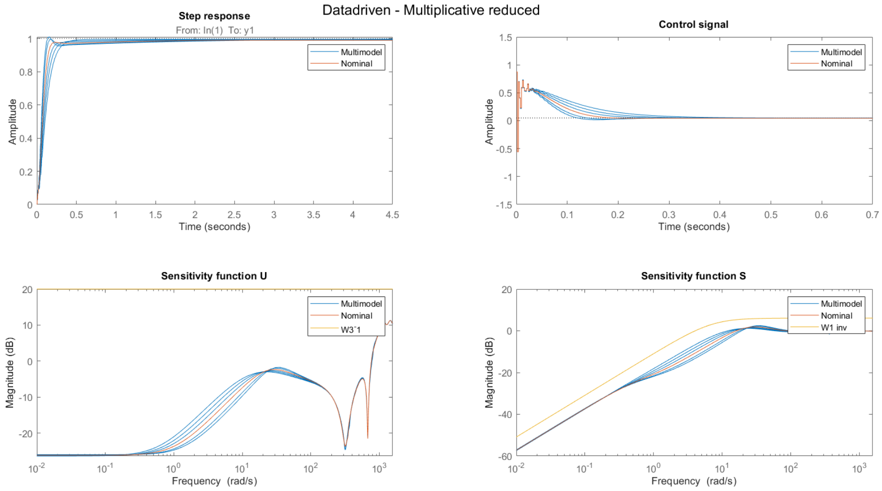

FIGURE 4.4
Data-driven controller using multiplicative uncertainty after reduction of the controller's order

Chapter 5

Data-driven controller Multimodel uncertainty

Here we want to compute a $\mathcal{H}_\infty$ controller by using the frequency-response of our system.
We will use the provided `datadriven.m` function to implement the data-driven approach.

Firstly we define the parameters that we are going to use for the controller identification:

- *The initial stabilizing controller*: We are taking as initial controller K_c a small gain integrator controller in the form: $K_c = \frac{c}{1-z^{-1}}$, where we have chosen $c = 0.001$, and verified that the resulting controller was stabilizing.
- *F_x and F_y* : Since $K = XY^{-1}$, observing our initial controller K_c we will choose as fixed part of the X, $F_x = z$ and as fixed part of Y, $F_y = z - 1$.
- *The required order of the final controller*: we will choose 11.
- *The frequency grid*: we have defined a frequency grid composed of 400 logarithmically spaced frequencies between 0 and π/T_s.

The minimization problem we want to solve has objective function:

$$\min \left\| \begin{bmatrix} W_1 \mathcal{S} \\ W_3 \mathcal{U} \end{bmatrix} \right\|_\infty \tag{5.1}$$

where $W_1(s)$ is the filter designed in Section 2.1, while $W_3(s)$ is the filter designed in Section 2.4.

The robust performance condition is:

$$\|W_1 S_i\| \leq 1 \quad \forall G_i$$

5.1

The resulting controller is:

```
K =

  0.9254 z^11 - 1.82 z^10 + 1.437 z^9 - 0.5439 z^8 - 0.2242 z^7 + 0.5703 z^6 - 0.2526 z^5 - 0.2735 z^4 + 0.3291 z^3 - 0.4543 z^2 + 0.3158 z - 0.008325
  --------------------------------------------------------------------------------------------------------------------------------------------------
  z^11 - 0.9899 z^10 - 0.1434 z^9 - 0.03623 z^8 - 0.0614 z^7 + 0.09748 z^6 - 0.003971 z^5 + 0.02552 z^4 + 0.4662 z^3 - 0.07621 z^2 - 0.1044 z - 0.1737
```

The performance can be observed in Fig. 5.1

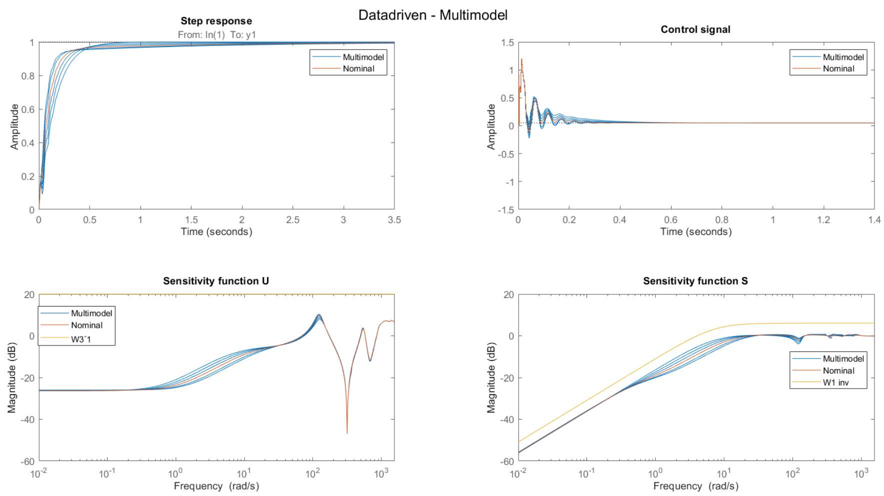

FIGURE 5.1
Data-driven controller using multimodel uncertainty

5.2

To compare with the other approach of design proposed in Chapter 2 we refer to Figure 5.3. We can observe that in the data-driven controller, the main difference is that the sensitivity function $\mathcal{U}$ does not have a peak around $\omega = 100$.

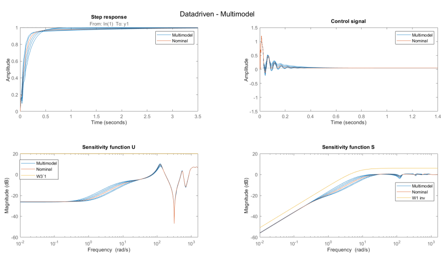

FIGURE 5.2
Data driven multimodel

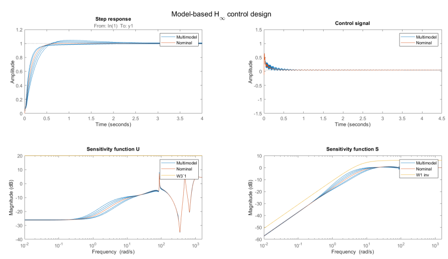

FIGURE 5.3
Mixed sensitivity Matlab

5.3

Now we want to design a reduced-order controller which has the same performance. We choose to reduce it to a 8-th order controller instead of 11-th order.

The performance can be checked in Fig. 5.4. We can see that the control signal is always inside the desired range and that the system is stable. Furthermore, the robust performance conditions are met such that

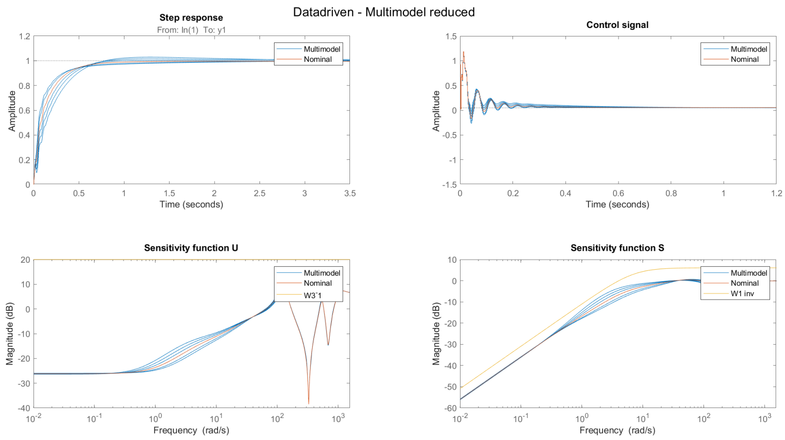

FIGURE 5.4
Data-driven controller using multimodel set after reduction of the controller's order

SUMMARY OF DATA-DRIVEN METHODS

ADVANTAGES

- Requires only frequency-response data, eliminating the need for a parametric model.
- Offers a range of design structures, such as low-order, centralized, decentralized, or distributed, using convex optimization.
- Integrates pure time delays (such as transportation or communication delays) in the design process.
- Facilitates mixed $\mathcal{H}_2$ and $\mathcal{H}_\infty$ control, suitable for diverse sensitivity functions and open-loop shaping.
- Directly addresses multimodel uncertainty, so there is no need of designing W_2.
- Provides a unified framework for designing both discrete- and continuous-time controllers.

DISADVANTAGES

- An initial stabilizing controller is required, which may not be immediately available.
- The selection of frequency data and frequency range requires careful consideration to ensure successful controller design.

Appendix A

MATLAB code

```
%% Computer exercise 2
clc, clear, close all

%% 2.1
% logs_1
load logs_1.mat
G1 = oe(data, [8 8 1]); % Parametric
Ts = G1.Ts; % Same for all data
freqs = (pi/4096:pi/4096:pi) / Ts; % Same for all data
Gf1 = spa(data,8191,freqs); % Non-parametric

opts = nyquistoptions;
opts.ConfidenceRegionDisplaySpacing = 3;
opts.ShowFullContour = 'off';

figure(1)
nyquistplot(Gf1,G1,freqs,opts,'sd',2.45);
legend('Non-parametric','Parametric')
% % add a zoomed zone
% box on
% zp = BaseZoom();
% zp.run;

% logs_3
load logs_3.mat
G3 = oe(data, [8 8 1]); % Parametric
Gf3 = spa(data,8191,freqs); % Non-parametric

opts = nyquistoptions;
opts.ConfidenceRegionDisplaySpacing = 3;
opts.ShowFullContour = 'off';

figure(2)
```

```
nyquistplot(Gf3,G3,freqs,opts,'sd',2.45);
legend('Non-parametric','Parametric')

% logs_5
load logs_5.mat
G5 = oe(data, [8 8 1]); % Parametric
Gf5 = spa(data,8191,freqs); % Non-parametric

opts = nyquistoptions;
opts.ConfidenceRegionDisplaySpacing = 3;
opts.ShowFullContour = 'off';

figure(3)
nyquistplot(Gf5,G5,freqs,opts,'sd',2.45);
legend('Non-parametric','Parametric')

% logs_7
load logs_7.mat
G7 = oe(data, [8 8 1]); % Parametric
Gf7 = spa(data,8191,freqs); % Non-parametric

opts = nyquistoptions;
opts.ConfidenceRegionDisplaySpacing = 3;
opts.ShowFullContour = 'off';

figure(4)
nyquistplot(Gf7,G7,freqs,opts,'sd',2.45);
legend('Non-parametric','Parametric')

% logs_9
load logs_9.mat
G9 = oe(data, [8 8 1]); % Parametric
Gf9 = spa(data,8191,freqs); % Non-parametric

opts = nyquistoptions;
opts.ConfidenceRegionDisplaySpacing = 3;
opts.ShowFullContour = 'off';

figure(5)
nyquistplot(Gf9,G9,freqs,opts,'sd',2.45);
legend('Non-parametric','Parametric')

% logs_11
load logs_11.mat
G11 = oe(data, [8 8 1]); % Parametric
Gf11 = spa(data,8191,freqs); % Non-parametric

opts = nyquistoptions;
opts.ConfidenceRegionDisplaySpacing = 3;
```

```
opts.ShowFullContour = 'off';

figure(6)
nyquistplot(Gf11,G11,freqs,opts,'sd',2.45);
legend('Non-parametric','Parametric')

% 2.1.5 To choose the nominal model we gonna plot the bode magnitude
    of the
% multimodel

% Bode diagram of all models
figure(7)
bodemag(G1,G3,G5,G7,G9,G11)
legend('G1','G3','G5','G7','G9','G11')

Gmm = stack(1,G1,G3,G5,G7,G9,G11);

% test all the nominal (by plotting the filter)
[Gu1, info1] = ucover(Gmm, G1, 2);
[Gu3, info3] = ucover(Gmm, G3, 2);
[Gu5, info5] = ucover(Gmm, G5, 2);
[Gu7, info7] = ucover(Gmm, G7, 2);
[Gu9, info9] = ucover(Gmm, G9, 2);
[Gu11,info11] = ucover(Gmm, G11, 2);

W2_1 = info1.W1opt;
W2_3 = info3.W1opt;
W2_5 = info5.W1opt;
W2_7 = info7.W1opt;
W2_9 = info9.W1opt;
W2_11 = info11.W1opt;

figure(8) % Gnom = G1
hold on
bodemag((G3/G1)-1, 'b')
bodemag((G5/G1)-1,'b')
bodemag((G7/G1)-1,'b')
bodemag((G9/G1)-1,'b')
bodemag((G11/G1)-1,'b')
bodemag(W2_1,'r')
title('Nominal model : G1')
legend('G3/G1)-1','(G5/G1)-1','G7/G1)-1','(G9/G1)-1','(G11/G1)-1','W2_1'
    )

figure(9) % Gnom = G3
hold on
bodemag((G1/G3)-1, 'b')
bodemag((G5/G3)-1,'b')
bodemag((G7/G3)-1,'b')
```

```
bodemag((G9/G3)-1,'b')
bodemag((G11/G3)-1,'b')
bodemag(W2_3,'r')
title('Nominal model : G3')

figure(10) % Gnom = G5
hold on
bodemag((G1/G5)-1, 'b')
bodemag((G3/G5)-1,'b')
bodemag((G7/G5)-1,'b')
bodemag((G9/G5)-1,'b')
bodemag((G11/G5)-1,'b')
bodemag(W2_5,'r')
title('Nominal model : G5')

figure(11) % Gnom = G7
hold on
bodemag((G1/G7)-1, 'b')
bodemag((G3/G7)-1,'b')
bodemag((G5/G7)-1,'b')
bodemag((G9/G7)-1,'b')
bodemag((G11/G7)-1,'b')
bodemag(W2_7,'r')
title('Nominal model : G7')

figure(12) % Gnom = G9
hold on
bodemag((G1/G9)-1, 'b')
bodemag((G3/G9)-1,'b')
bodemag((G5/G9)-1,'b')
bodemag((G7/G9)-1,'b')
bodemag((G11/G9)-1,'b')
bodemag(W2_9,'r')
title('Nominal model : G9')

figure(13) % Gnom = G11
hold on
bodemag((G1/G11)-1, 'b')
bodemag((G3/G11)-1,'b')
bodemag((G5/G11)-1,'b')
bodemag((G7/G11)-1,'b')
bodemag((G9/G11)-1,'b')
bodemag(W2_11,'r')
title('Nominal model : G11')

% All W2 in one plot
figure(14)
bodemag(W2_1,W2_3,W2_5,W2_7,W2_9,W2_11)
title('Comparison between different weighting filter')
```

```
legend('W2 (Gnom = G1)','W2 (Gnom = G3)','W2 (Gnom = G5)', ...
    'W2 (Gnom = G7)','W2 (Gnom = G9)','W2 (Gnom =
       G11)','Location','northwest')

[Gu74, info74] = ucover(Gmm, G7, 4);
W2 = info74.W1;
Gnom = G7;
save('W2')
save('Gnom')
save('Gmm')

%% 2.2
% Load data
load('Gnom')
load('Gmm')
load('W2')

% Robust performance condition
% W1S < 1 for Gmm
% [W1S W2T] < sqrt(2)/2 for Gnom

% Design of W1
num = [1 7];
den = [1 0.0001];
W1 = tf(num,den) * 1/2;
W1 = c2d(W1,Ts,'zoh');
figure(15)
bodemag(W1^-1,tf(2))

% Design of K (Hinf controller)
Kinf = mixsyn(Gnom,W1,[],W2);

T = feedback(Gmm*Kinf,1);
Tnom = feedback(Gnom*Kinf,1);
stepinfo(Tnom) % Check settling time
U =feedback(Kinf,Gmm);
Unom =feedback(Kinf,Gnom);
S = feedback(1,Gmm*Kinf);
Snom = feedback(1,Gnom*Kinf);

% Plot
figure(16)
subplot(2,2,1)
step(T,Tnom)
title('Step response')
legend('Multimodel','Nominal')
subplot(2,2,2)
step(U,Unom)
title('Control signal')
```

```
legend('Multimodel','Nominal')
ylim([-1.5,1.5])
subplot(2,2,3)
bodemag(U,Unom)
xlim([10^-2 1560])
title('Sensitvity function U')
legend('Multimodel','Nominal')
subplot(2,2,4)
bodemag(S,Snom, W1^-1)
xlim([10^-2 1560])
title('Sensitivity function S')
legend('Multimodel','Nominal', 'W1 inv')

condition = norm([W1*Snom W2*Tnom],inf)
if (condition <= 1/sqrt(2))
    fprintf('Robust performance conditions are met\n')
end

% W3 to imit the magnitude of U
W3 = 0.1;
Kinf_range = mixsyn(Gnom, W1, W3, W2);

T = feedback(Gmm*Kinf_range,1);
Tnom = feedback(Gnom*Kinf_range,1);
U =feedback(Kinf_range,Gmm);
Unom =feedback(Kinf_range,Gnom);
S = feedback(1,Gmm*Kinf_range);
Snom = feedback(1,Gnom*Kinf_range);

% Plot
figure(17)
subplot(2,2,1)
step(T,Tnom)
title('Step response')
legend('Multimodel','Nominal')
subplot(2,2,2)
step(U,Unom)
title('Control signal')
legend('Multimodel','Nominal')
ylim([-1.5,1.5])
subplot(2,2,3)
bodemag(U,Unom,tf(1/W3))
title('Sensitivity function U')
legend('Multimodel','Nominal','W3^-1')
xlim([10^-2 1560])
subplot(2,2,4)
bodemag(S,Snom, W1^-1)
title('Sensitivity function S')
xlim([10^-2 1560])
```

```
legend('Multimodel','Nominal', 'W1 inv')

condition = norm([W1*Snom W2*Tnom],inf)

if (condition <= 1/sqrt(2))
    fprintf('Robust performance conditions are met\n')
end

% Reduce order of K
orderK = size(Kinf_range.A,1);

K_ft = ss2tf(Kinf_range.A,Kinf_range.B,Kinf_range.C,Kinf_range.D);
Kreduced = reduce(Kinf_range, 11); % 11th-order
orderKreduced = size(Kreduced.A,1);

% Comparison pole-zero map
figure(18)
subplot(1,2,2)
pzmap(Kreduced)
title('Controller after reduction')
subplot(1,2,1)
pzmap(Kinf_range)
title('Controller before reduction')

% Comparison controllers
figure(19)
bodemag(Kinf_range,Kreduced)
legend('Controller before reduction', 'Controller after reduction')
title('Reduction of the order of the controller')

% Test performance
T = feedback(Gmm*Kreduced,1);
Tnom = feedback(Gnom*Kreduced,1);
U =feedback(Kreduced,Gmm);
Unom =feedback(Kreduced,Gnom);
S = feedback(1,Gmm*Kreduced);
Snom = feedback(1,Gnom*Kreduced);

figure(20)
subplot(2,2,1)
step(T,Tnom)
title('Step response')
legend('Multimodel','Nominal')
subplot(2,2,2)
step(U,Unom)
title('Control signal')
legend('Multimodel','Nominal')
ylim([-1.5,1.5])
subplot(2,2,3)
```

```
bodemag(U,Unom, 1/tf(W3))
xlim([10^-2 1560])
title('Sensitivity function U')
legend('Multimodel','Nominal','W3^-1')
subplot(2,2,4)
bodemag(S,Snom, W1^-1)
xlim([10^-2 1560])
title('Sensitivity function S')
sgtitle('Model-based H_\infty control design')
legend('Multimodel','Nominal', 'W1 inv')

condition1 = norm([W1*Snom W2*Tnom],inf)
if (condition <= 1/sqrt(2))
    fprintf('Robust performance conditions is met\n')
end

% Infinity norm
norm_nominal = norm([W1*Snom W2*Tnom],inf)
norm_multimodel = norm([W1*S W2*T],inf)

% Modify report + report Datadriven
%% H2 controller
% Conversion from discrete time to continuous time
Gct = d2c(Gnom);
[A,B,C,D] = ssdata(Gct);

% Optimization problem definition
n = size(A,1);
m = size(B,2);

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

obj = trace(C*L*C') + trace(M); % Objective function

% LMI definitions
lmi1 = A*L -B*X + L*A' - X'*B' + B*B' <= 0;
lmi2 = [M X; X' L] >= 0;
lmi3 = L >= 0;
lmi = [lmi1,lmi2,lmi3];

% Options
options = sdpsettings('solver','mosek');
optimize(lmi,obj,options);

% Controller
X = value(X);
```

```
L = value(L);
K_H2 = X*inv(L);

% Step response of closed loop system
Acl = A - B*K_H2;
Bcl = B;
Ccl = C;
Dcl = D;
sys_cl = ss(Acl,Bcl,Ccl,Dcl);

figure(21)
step(sys_cl)
title('Step response using K_{H2} controller')

% Step response with LQR
Q = C'*C;
R = eye(m);
[K_lqr,~,~] = lqr(A,B,Q,R);
Acl_lqr = A - B*K_lqr;
sys_cl_lqr = ss(Acl_lqr,Bcl,Ccl,Dcl);

% Comparison
figure(22)
step(sys_cl_lqr, '*')
hold on
step(sys_cl, '-g')
legend('Step response using LQR', 'Step response using H2')
title('Step response comparison')

%% Data-driven controller - multimodel
% Load empty structure
[SYS, OBJ, CON, PAR] = datadriven.utils.emptyStruct();

% ------------------------------------------------------------------------------
%   Initial controller
% ------------------------------------------------------------------------------
z = tf('z',Ts);
c = 0.001;
Kc = c / (1 - z^-1);                      % Initial controller
[num, den] = tfdata(Kc, 'v');    % Extract numerator and denominator

order = orderKreduced;
den(order + 1) = 0; % Zero padding to have same order as desired controller
num(order + 1) = 0; % Zero padding to have same order as desired controller

```

```matlab
% Fixed parts of the controller
%   NOTE: Initial controller should contain the fixed parts too!
Fy = [1 -1];                     % Fixed part of denominator as polynomial
den = deconv(den, Fy);   % Remove fixed part of denominator

Fx = 1;                  % Fixed part of numerator as polynomial
num = deconv(num, Fx);   % Remove fixed part of numerator

SYS.controller.num = num;
SYS.controller.den = den;
SYS.controller.Ts = Ts;
SYS.controller.Fx = Fx;
SYS.controller.Fy = Fy;

% ------------------------------------------------------------------------------------
%   Nominal system(s)
% ------------------------------------------------------------------------------------
% Systems should be LTI systems ('ss', 'tf', 'frd', ...)
SYS.model = Gmm;

% ------------------------------------------------------------------------------------
%   Frequencies for controller synthesis
% ------------------------------------------------------------------------------------
SYS.W = logspace(0,log10(pi/Ts),400);

% ------------------------------------------------------------------------------------
%   Filters for objectives
% ------------------------------------------------------------------------------------
% Filter should be LTI systems ('ss', 'tf', 'frd', ...)
% For unused objectives, set filters to []
OBJ.oinf.W1 = W1;   %   W1     S
OBJ.oinf.W2 = [];   %   W2     T
OBJ.oinf.W3 = tf(W3);   %   W3     U
OBJ.oinf.W4 = [];   %   W4     V

OBJ.o2.W1 = [];     %   W1     S
OBJ.o2.W2 = [];     %   W2     T
OBJ.o2.W3 = [];     %   W3     U
OBJ.o2.W4 = [];     %   W4     V

OBJ.LSinf.Ld = [];  %   W   (Ld - G K)
OBJ.LSinf.W  = [];
```

```

OBJ.LS2.Ld = [];     %   W   (Ld - G K)
OBJ.LS2.W  = [];

% ------------------------------------------------------------------------------------------------
%   Filters for constraints
% ------------------------------------------------------------------------------------------------
% Filter should be LTI systems (`ss`, `tf`, `frd`, ...)
% For unused constraints, set filters to []
CON.W1 = [];     %   W1     S          1
CON.W2 = [];     %   W2     T          1
CON.W3 = [];     %   W3     U          1
CON.W4 = [];     %   W4     V          1

% ------------------------------------------------------------------------------------------------
%   Optimisation parameters
% ------------------------------------------------------------------------------------------------
PAR.tol = 1e-4;      % Numerical tolerance for convergence
PAR.maxIter = 100;   % Maximum number of allowed iterations

verbosity = true;    % To print controller synthesis iterations
solver = "mosek";         % Solver to use for optimisation ("mosek", "sedumi", ...)

% ------------------------------------------------------------------------------------------------
%   Solve the datadriven controller synthesis problem
% ------------------------------------------------------------------------------------------------
[K, sol] = datadriven.datadriven(SYS, OBJ, CON, PAR, verbosity, solver);

% Test performance
T = feedback(Gmm*K,1);
Tnom = feedback(Gnom*K,1);
U = feedback(K,Gmm);
Unom = feedback(K,Gnom);
S = feedback(1,Gmm*K);
Snom = feedback(1,Gnom*K);

figure(21)
subplot(2,2,1)
step(T,Tnom)
title('Step response')
legend('Multimodel','Nominal')
```

```
subplot(2,2,2)
step(U,Unom)
title('Control signal')
legend('Multimodel','Nominal')
ylim([-1.5,1.5])
subplot(2,2,3)
bodemag(U,Unom, 1/tf(W3))
xlim([10^-2 1560])
title('Sensitivity function U')
legend('Multimodel','Nominal','W3^-1')
subplot(2,2,4)
bodemag(S,Snom, W1^-1)
xlim([10^-2 1560])
title('Sensitivity function S')
legend('Multimodel','Nominal', 'W1 inv')
sgtitle('Datadriven - Multimodel')
saveas(gca,'CE2_datadriven_multimodel','png')

condition = norm(W1*S,inf);

if all(condition <= 1)
    fprintf('Robust performance conditions are met\n')
end

% Reduction of the controller
Kred = reduce(K,8);

figure(23)
bodemag(K,Kred)
legend('Controller before reduction', 'Controller after reduction')
title('Reduction of the order of the controller')

% Test performance
T = feedback(Gmm*Kred,1);
Tnom = feedback(Gnom*Kred,1);
U = feedback(Kred,Gmm);
Unom = feedback(Kred,Gnom);
S = feedback(1,Gmm*Kred);
Snom = feedback(1,Gnom*Kred);

figure(24)
subplot(2,2,1)
step(T,Tnom)
title('Step response')
legend('Multimodel','Nominal')
subplot(2,2,2)
step(U,Unom)
title('Control signal')
legend('Multimodel','Nominal')
```

```
ylim([-1.5,1.5])
subplot(2,2,3)
bodemag(U,Unom, 1/tf(W3))
xlim([10^-2 1560])
title('Sensitivity function U')
legend('Multimodel','Nominal','W3^-1')
subplot(2,2,4)
bodemag(S,Snom, W1^-1)
xlim([10^-2 1560])
title('Sensitivity function S')
legend('Multimodel','Nominal', 'W1 inv')
sgtitle('Datadriven - Multimodel reduced')
saveas(gca,'CE2_datadriven_multimodel_reduced','png')

condition = norm(W1*S,inf)

if all(condition <= 1)
    fprintf('Robust performance conditions are met\n')
end
%% Data-driven controller - multiplicative
% Load empty structure
[SYS, OBJ, CON, PAR] = datadriven.utils.emptyStruct();

%
    -----------------------------------------------------------------------------
%   Initial controller
%
    -----------------------------------------------------------------------------
z = tf('z',Ts);
c = 0.001;
Kc = c / (1 - z^-1);                           % Initial controller
[num, den] = tfdata(Kc, 'v');    % Extract numerator and denominator

order = orderKreduced;
den(order + 1) = 0; % Zero padding to have same order as desired
    controller
num(order + 1) = 0; % Zero padding to have same order as desired
    controller

% Fixed parts of the controller
%   NOTE: Initial controller should contain the fixed parts too!
Fy = [1 -1];                       % Fixed part of denominator as
    polynomial
den = deconv(den, Fy);   % Remove fixed part of denominator

Fx = 1;                  % Fixed part of numerator as polynomial
num = deconv(num, Fx);   % Remove fixed part of numerator

SYS.controller.num = num;
```

```
SYS.controller.den = den;
SYS.controller.Ts = Ts;
SYS.controller.Fx = Fx;
SYS.controller.Fy = Fy;

% ---------------------------------------------------------------------------------
%   Nominal system(s)
% ---------------------------------------------------------------------------------
% Systems should be LTI systems ('ss', 'tf', 'frd', ...)
SYS.model = Gnom;

% ---------------------------------------------------------------------------------
%   Frequencies for controller synthesis
% ---------------------------------------------------------------------------------
SYS.W = logspace(0,log10(pi/Ts),400);

% ---------------------------------------------------------------------------------
%   Filters for objectives
% ---------------------------------------------------------------------------------
% Filter should be LTI systems ('ss', 'tf', 'frd', ...)
% For unused objectives, set filters to []
OBJ.oinf.W1 = W1;    %   W1     S
OBJ.oinf.W2 = W2;    %   W2     T
OBJ.oinf.W3 = tf(W3);    %   W3     U
OBJ.oinf.W4 = [];    %   W4     V

OBJ.o2.W1 = [];      %   W1     S
OBJ.o2.W2 = [];      %   W2     T
OBJ.o2.W3 = [];      %   W3     U
OBJ.o2.W4 = [];      %   W4     V

OBJ.LSinf.Ld = [];  %   W   (Ld - G K)
OBJ.LSinf.W  = [];

OBJ.LS2.Ld = [];     %   W   (Ld - G K)
OBJ.LS2.W  = [];

% ---------------------------------------------------------------------------------
%   Filters for constraints
% ---------------------------------------------------------------------------------
% Filter should be LTI systems ('ss', 'tf', 'frd', ...)
```

```
% For unused constraints, set filters to []
CON.W1 = [];     %   W1     S          1
CON.W2 = [];     %   W2     T          1
CON.W3 = [];     %   W3     U          1
CON.W4 = [];     %   W4     V          1

%
    ---------------------------------------------------------------------------------------
%   Optimisation parameters
%
    ---------------------------------------------------------------------------------------
PAR.tol = 1e-4;      % Numerical tolerance for convergence
PAR.maxIter = 100;   % Maximum number of allowed iterations

verbosity = true;    % To print controller synthesis iterations
solver = "mosek";         % Solver to use for optimisation ("mosek",
    "sedumi", ...)

%
    ---------------------------------------------------------------------------------------
%   Solve the datadriven controller synthesis problem
%
    ---------------------------------------------------------------------------------------
[K, sol] = datadriven.datadriven(SYS, OBJ, CON, PAR, verbosity,
    solver);

% Test performance
T = feedback(Gmm*K,1);
Tnom = feedback(Gnom*K,1);
U = feedback(K,Gmm);
Unom = feedback(K,Gnom);
S = feedback(1,Gmm*K);
Snom = feedback(1,Gnom*K);

figure(22)
subplot(2,2,1)
step(T,Tnom)
title('Step response')
legend('Multimodel','Nominal')
subplot(2,2,2)
step(U,Unom)
title('Control signal')
legend('Multimodel','Nominal')
ylim([-1.5,1.5])
subplot(2,2,3)
bodemag(U,Unom, 1/tf(W3))
xlim([10^-2 1560])
title('Sensitivity function U')
legend('Multimodel','Nominal','W3^-1')
```

```
subplot(2,2,4)
bodemag(S,Snom, W1^-1)
xlim([10^-2 1560])
title('Sensitivity function S')
legend('Multimodel','Nominal', 'W1 inv')
sgtitle('Datadriven - Multiplicative')
saveas(gca,'CE2_datadriven_multiplicative','png')

condition = norm([W1*Snom W2*Tnom],inf);

if condition <= 1/sqrt(2)
    fprintf('Robust performance conditions are met\n')
end

% Reduction of the controller
Kred = reduce(K,8);

figure(23)
bodemag(K,Kred)
legend('Controller before reduction', 'Controller after reduction')
title('Reduction of the order of the controller')

% Test performance
T = feedback(Gmm*Kred,1);
Tnom = feedback(Gnom*Kred,1);
U = feedback(Kred,Gmm);
Unom = feedback(Kred,Gnom);
S = feedback(1,Gmm*Kred);
Snom = feedback(1,Gnom*Kred);

figure(24)
subplot(2,2,1)
step(T,Tnom)
title('Step response')
legend('Multimodel','Nominal')
subplot(2,2,2)
step(U,Unom)
title('Control signal')
legend('Multimodel','Nominal')
ylim([-1.5,1.5])
subplot(2,2,3)
bodemag(U,Unom, 1/tf(W3))
xlim([10^-2 1560])
title('Sensitivity function U')
legend('Multimodel','Nominal','W3^-1')
subplot(2,2,4)
bodemag(S,Snom, W1^-1)
xlim([10^-2 1560])
title('Sensitivity function S')
```

```
legend('Multimodel','Nominal', 'W1 inv')
sgtitle('Datadriven - Multiplicative reduced')
saveas(gca,'CE2_datadriven_multiplicative_reduced','png')

condition = norm(W1*S,inf);

if all(condition <= 1)
    fprintf('Robust performance conditions are met\n')
end
```