



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

<b>1</b>	<b>Multiplicative uncertainty</b>	<b>2</b>
1.1	.....	2
1.2	.....	3
1.3	.....	3
1.4	.....	4
1.5	.....	4
1.6	.....	5
<b>2</b>	<b>Model-Based <math>\mathcal{H}_\infty</math> Controller Design</b>	<b>7</b>
2.1	.....	7
2.2	.....	8
2.3	.....	9
2.4	.....	10
2.5	.....	11
2.6	.....	11
<b>3</b>	<b>Model-Based <math>\mathcal{H}_2</math> Controller Design</b>	<b>13</b>
3.1	.....	13
3.2	.....	13
3.3	.....	14
3.4	.....	15
3.5	.....	15
3.6	.....	16
<b>4</b>	<b>Data-driven controller Multiplicative uncertainty</b>	<b>18</b>
4.1	.....	18
4.2	.....	19
4.3	.....	19
<b>5</b>	<b>Data-driven controller Multimodel uncertainty</b>	<b>21</b>
5.1	.....	21
5.2	.....	22
5.3	.....	22
<b>A</b>	<b>MATLAB code</b>	<b>24</b>

# 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

```
1 load logs_1.mat
2 G1 = oe(data, [8 8 1]); % Parametric
3 Ts = G1.Ts; % Same for all data
4 freqs = (pi/4096:pi/4096:pi) / Ts; % Same for all data
5 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

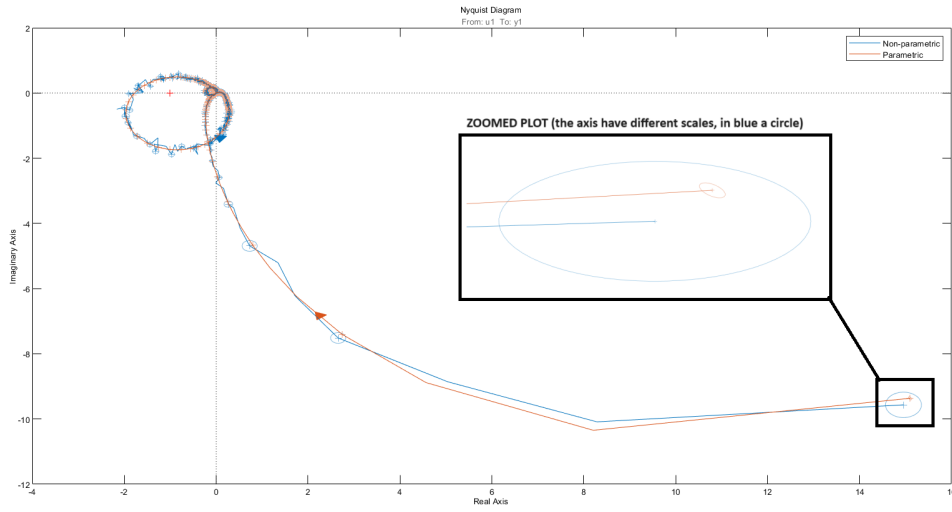
```
1 opts = nyquistoptions;
2 opts.ConfidenceRegionDisplaySpacing = 3;
3 opts.ShowFullContour = 'off';
4
5 figure(1)
6 nyquistplot(Gf1,G1,freqs,opts,'sd',2.45);
7 legend('Non-parametric','Parametric')
```

## 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

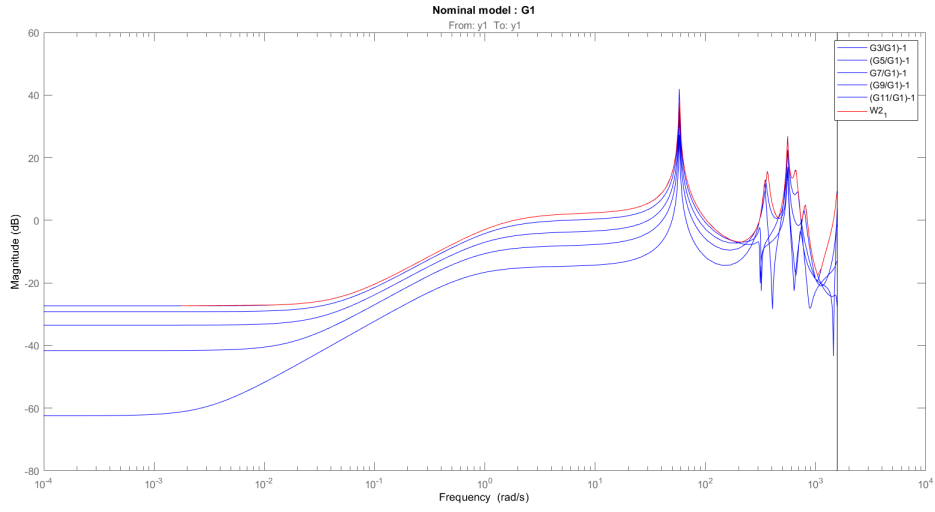
```
1 Gmm = stack(1, G1, G3, G5, G7, G9, G11);
```

## 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:

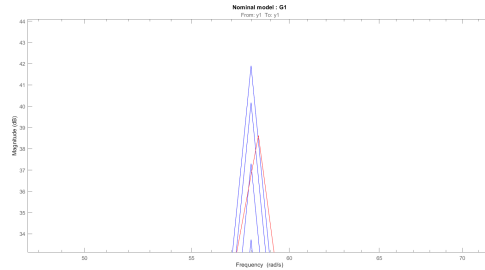
**LISTING 1.4**  
W2 with G1 as nominal system

```
1 [Gu1, info1] = ucover(Gmm, G1, 2);
2 W2_1 = info1.Wlopt;
3 figure(8) % Gnom = G1
4 hold on
5 bodemag((G3/G1)-1, 'b')
6 bodemag((G5/G1)-1, 'b')
7 bodemag((G7/G1)-1, 'b')
8 bodemag((G9/G1)-1, 'b')
9 bodemag((G11/G1)-1, 'b')
10 bodemag(W2_1, 'r')
11 title('Nominal model : G1')
12 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.

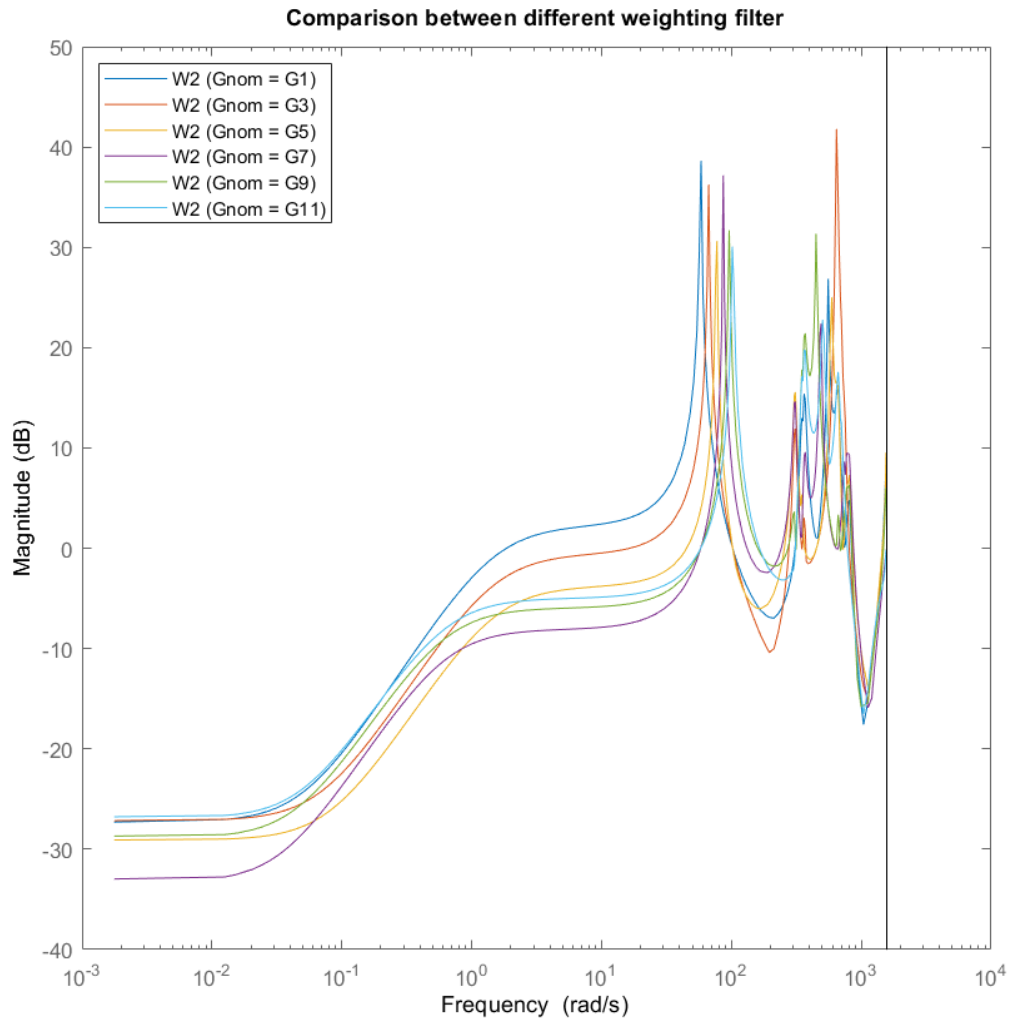


**FIGURE 1.4**  
 $W_2$  is cutting the peak

## 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  $10^3 \text{ rad/s}$ ) 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  $\infty$ -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  $W_1(z)$ :

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

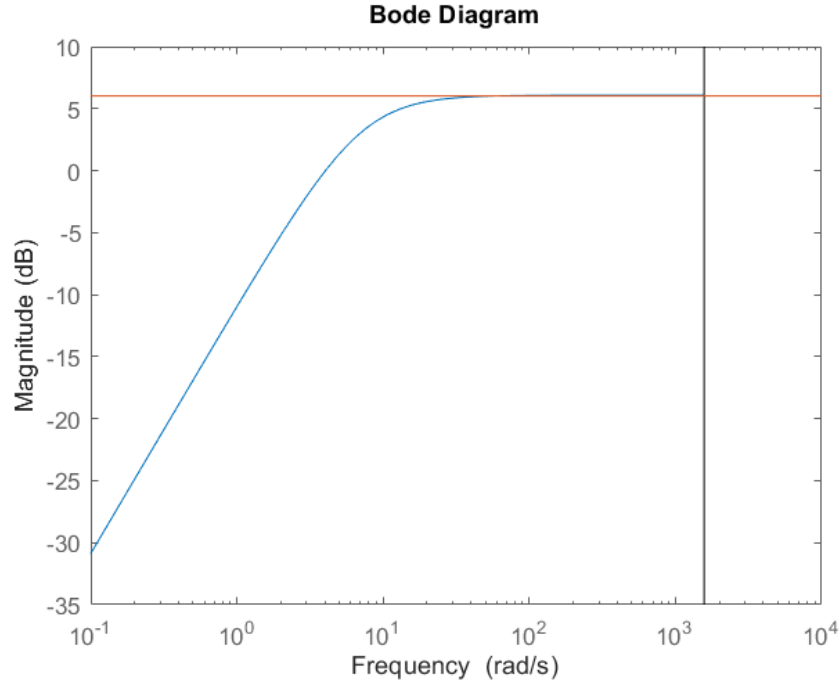
The settling time using this filter is 0.49s

**LISTING 2.1**  
Design of W1

```
1 num = [1 7];  
2 den = [1 0.0001];  
3 W1 = tf(num,den) * 1/2;  
4 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  $K_{H_\infty}$  for robust performance. We use the mixed sensitivity approach with the MATLAB function `mixsyn`. This function provide a controller that minimize the  $H_\infty$  norm of the weighted closed-loop transfer function :

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

where we defined :

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

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  $T$ , 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  $W_2$

```

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

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

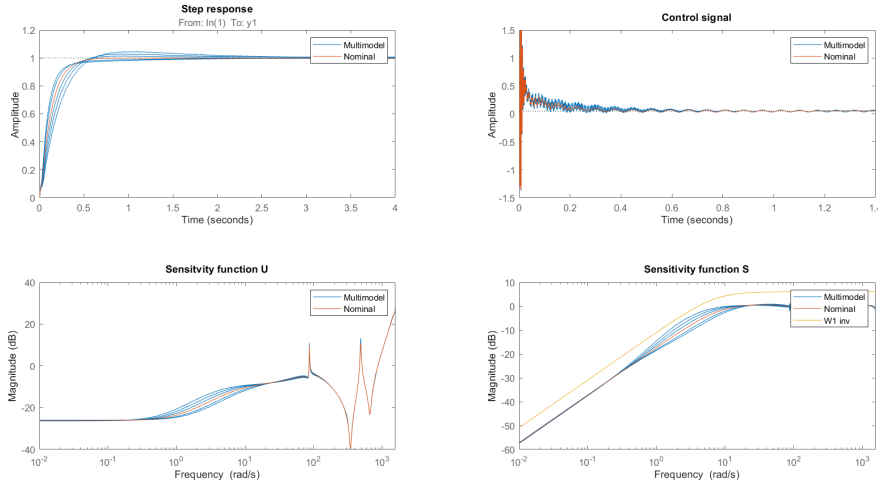


FIGURE 2.2

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

$\mathcal{H}_\infty$  controller

```
1 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  $\mathcal{S}, \mathcal{T}, \mathcal{U}$

```
1 T = feedback(Gmm*Kinf,1);
2 Tnom = feedback(Gnom*Kinf,1);
3 U =feedback(Kinf,Gmm);
4 Unom =feedback(Kinf,Gnom);
5 S = feedback(1,Gmm*Kinf);
6 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

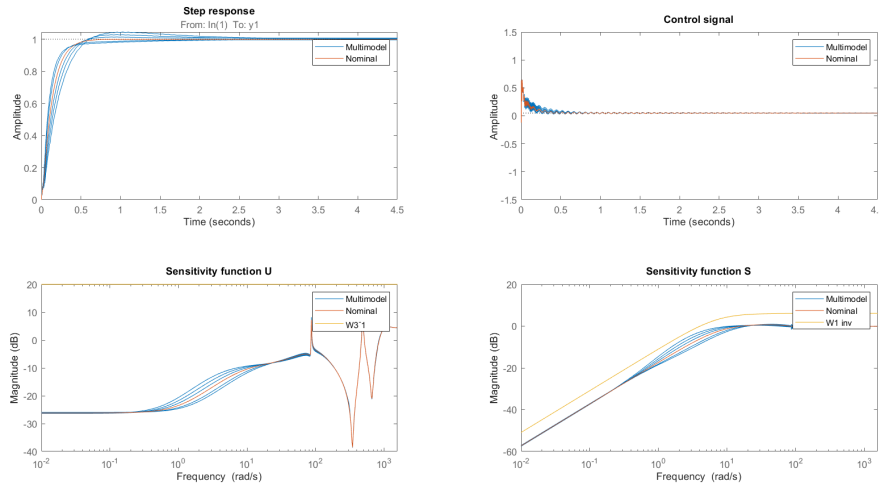
Robust performance conditions check

```

1 condition = norm([W1*Snom W2*Tnom],inf)
2 if (condition <= 1/sqrt(2))
3     fprintf('Robust performance conditions are met\n')
4 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  $\pm 1.5V$ . 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  $\pm 1.5V$

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  $W_3$  and redefinition of the controller

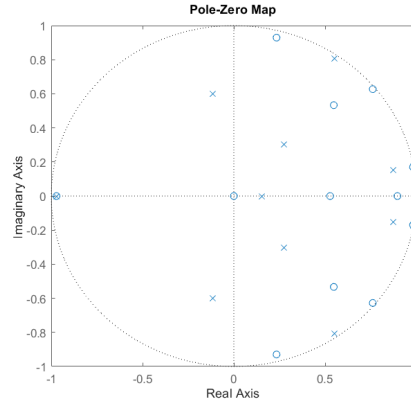
```

1 W3 = 0.1;
2 Kinf_range = mixsyn(Gnom, W1, W3, W2);
3
4 T = feedback(Gmm*Kinf_range,1);
5 Tnom = feedback(Gnom*Kinf_range,1);
6 U =feedback(Kinf_range,Gmm);
7 Unom =feedback(Kinf_range,Gnom);
8 S = feedback(1,Gmm*Kinf_range);
    
```

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

## 2.5

In this section we want to possibly reduce the order of our controller  $K_{H_\infty}$ . 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

```
1 | K_ft = ss2tf(Kinf_range.A,Kinf_range.B,Kinf_range.C,Kinf_range.D);
2 | 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 :

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

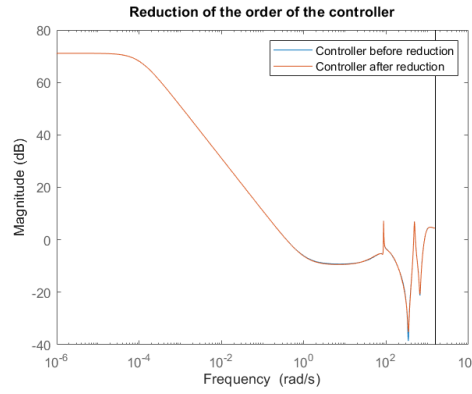
## 2.6

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

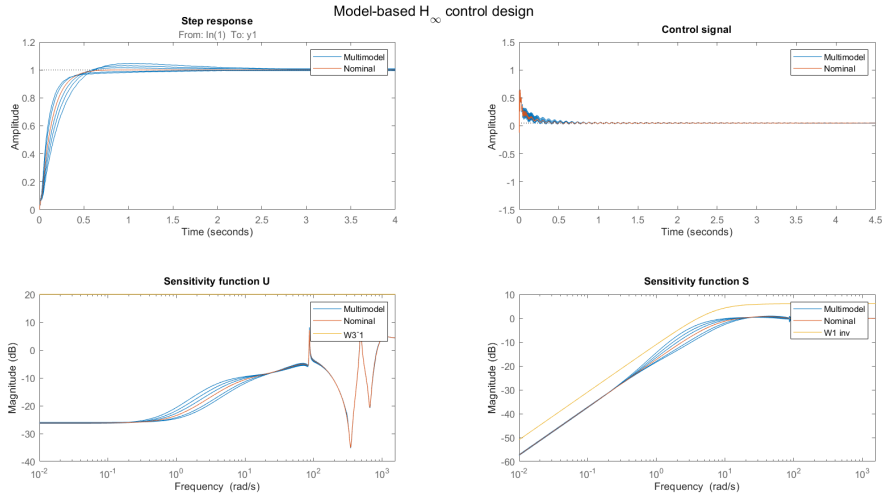
$$\left\| \begin{bmatrix} W_1 S \\ W_2 \mathcal{T} \end{bmatrix} \right\|_\infty$$

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

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  $U$  and  $S$  with a range for the control signal of  $\pm 1.5V$ . The controller is reduced to the 11th-order

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

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  $\infty$ -norm

```
1 norm_nominal = norm([W1*Snom W2*Tnom],inf)
2 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

```
1 % Conversion from discrete time to continuous time
2 Gct = d2c(Gnom);
3 [A,B,C,D] = ssdata(Gct);
```

### 3.2

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

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

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

$$u(t) = -Kx(t) + v(t) \quad (3.2)$$

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:

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

Hence in our case we want to minimize:

$$\|G_{v \rightarrow y_1}\|_2^2 + \|G_{v \rightarrow y_2}\|_2^2 \quad (3.4)$$

### 3.3

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

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

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:

$$AL + LA^T - BX - X^T B^T + BB^T \preceq 0 \quad (3.5)$$

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

For the objective function, we can define a new variable  $M \succeq K L K^T$ , 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:

$$M - X L X^{-1} \succeq 0 \quad (3.6)$$

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

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

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

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

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

**LISTING 3.2**  
2-Norm controller

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

```

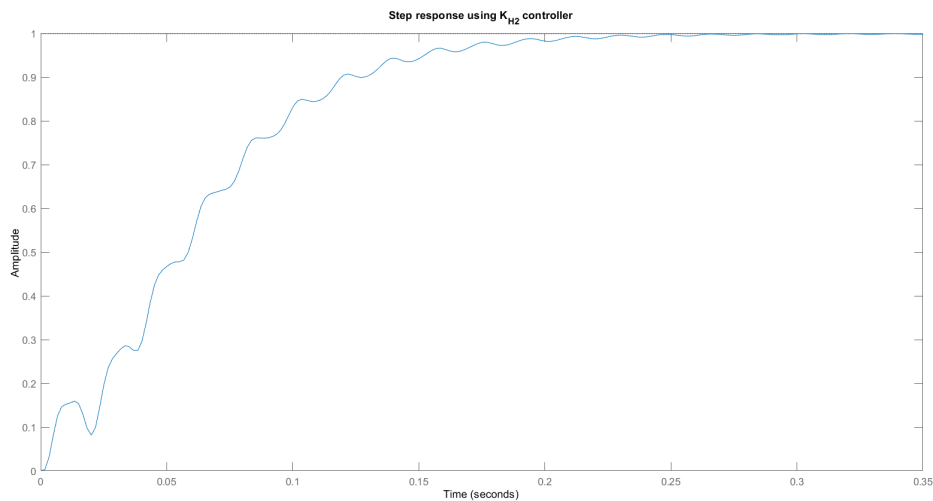
3 m = size(B,2);
4 % decision variables:
5 L = sdpvar(n,n,'symmetric');
6 X = sdpvar(m,n);
7 M = sdpvar(m,m);
8 %objective function:
9 obj = trace(C*L*C') + trace(M);
10 % lmi definition:
11 lmi1 = A*L - B*X + L*A' - X'*B' + B*B' <= 0;
12 lmi2 = [M X; X' L] >= 0;
13 lmi3 = L >= 0;
14 lmi = [lmi1,lmi2,lmi3];
15 % options:
16 options = sdpsettings('solver','mosek');
17 optimize(lmi,obj,options);
18 % controller:
19 X = value(X);
20 L = value(L);
21 K_H2 = X*inv(L);
    
```

### 3.4

The resulting controller is:

$$K_{H_2} = [2.0000 \quad 0.4656 \quad 2.8658 \quad 1.2251 \quad 2.4430 \quad 1.7477 \quad 1.4860 \quad 2.6511 \quad 9.8458] \quad (3.9)$$

### 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:

$$J(u) = \int_0^\infty (x^T Q x + u^T R u) dt \quad (3.10)$$

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

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

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:

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

which can be rewritten as:

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

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

$$\|G_{v \rightarrow y_1}\|_2^2 + \|G_{v \rightarrow y_2}\|_2^2 \quad (3.14)$$

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:

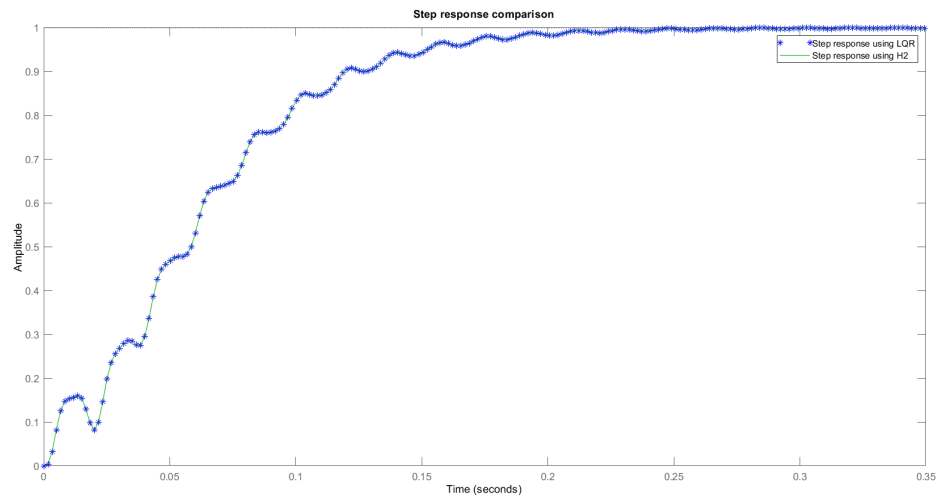
#### LISTING 3.3 LQR

```
1 % LQR
2 Q = C' * C;
3 R = eye(m);
4 [K_lqr, ~, ~] = lqr(A, B, Q, R);
```

The resulting controller is:

$$K_{LQR} = [2.0000 \quad 0.4656 \quad 2.8658 \quad 1.2251 \quad 2.4430 \quad 1.7477 \quad 1.4860 \quad 2.6511 \quad 9.8458] \quad (3.15)$$

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  $\pi/T_s$ .

The minimization problem we want to solve has objective function:

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

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 =

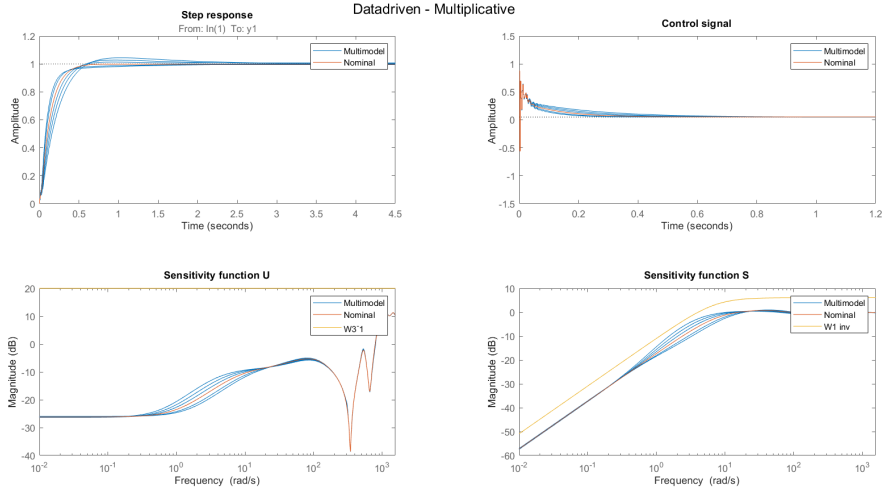
$$\frac{0.8729 z^{11} - 2.043 z^{10} + 1.895 z^9 - 0.5845 z^8 - 0.5 z^7 + 0.4555 z^6 + 0.1278 z^5 - 0.3003 z^4 - 0.2262 z^3 + 0.435 z^2 - 0.1904 z + 0.0593}{z^{11} - 0.7072 z^{10} - 0.4374 z^9 + 0.03228 z^8 + 0.02915 z^7 - 0.1737 z^6 - 0.07207 z^5 + 0.2271 z^4 + 0.1325 z^3 + 0.1617 z^2 - 0.05927 z -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

## 4.2

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

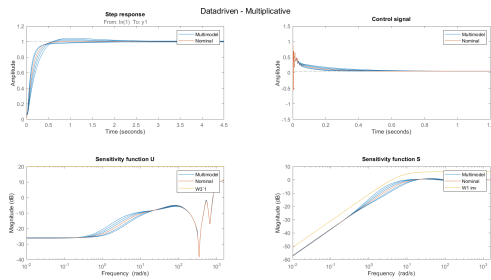
$$\left\| \begin{bmatrix} W_1 \mathcal{S} \\ W_2 \mathcal{T} \end{bmatrix} \right\|_{\infty} = 0.6130 < \frac{1}{\sqrt{2}} \quad (4.2)$$

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

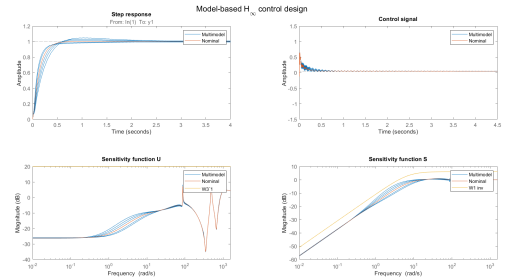


**FIGURE 4.1**  
Data driven Multiplicative uncertainty

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

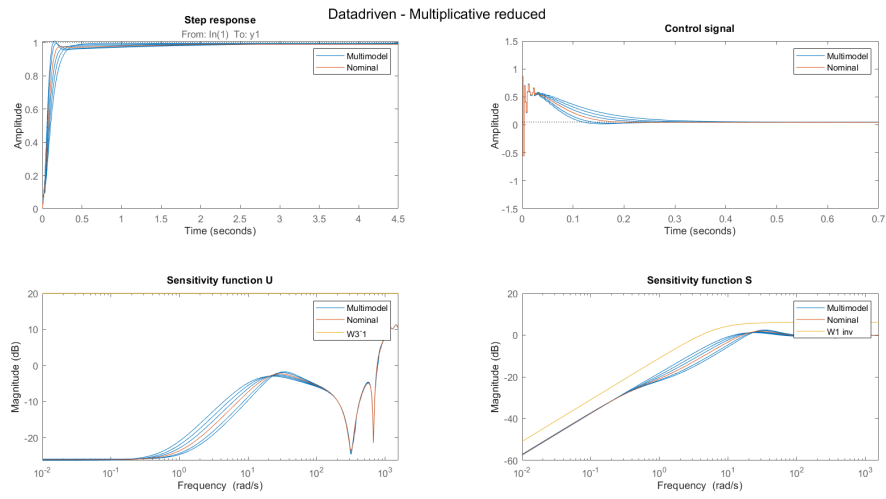


**FIGURE 4.3**  
Mixed sensitivity Matlab

## 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  $\pi/T_s$ .

The minimization problem we want to solve has objective function:

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

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:

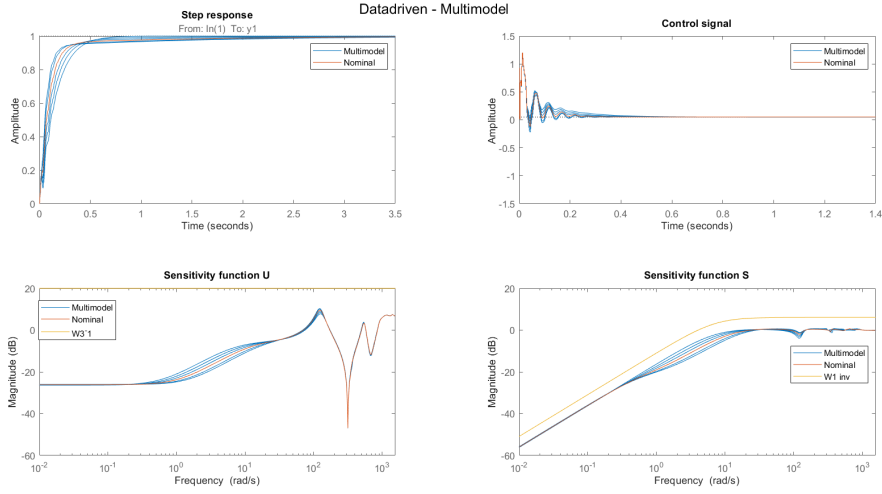
$$\|W_1 S_i\| \leq 1 \quad \forall G_i$$

### 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
```

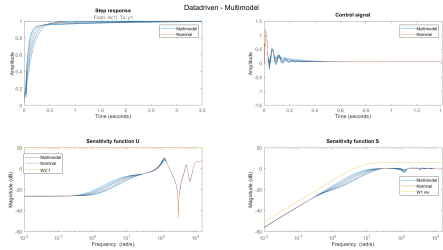
The performance can be observed in Fig. 5.1



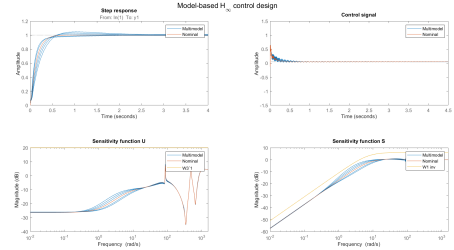
**FIGURE 5.1**  
Data-driven controller using multimodel uncertainty

## 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

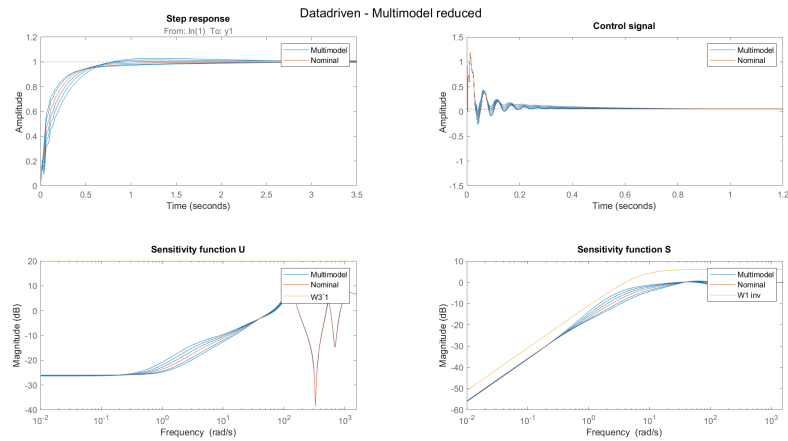


FIGURE 5.4

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

```
1 %% Computer exercise 2
2 clc, clear, close all
3
4 %% 2.1
5 % logs_1
6 load logs_1.mat
7 G1 = oe(data, [8 8 1]); % Parametric
8 Ts = G1.Ts; % Same for all data
9 freqs = (pi/4096:pi/4096:pi) / Ts; % Same for all data
10 Gf1 = spa(data,8191,freqs); % Non-parametric
11
12 opts = nyquistoptions;
13 opts.ConfidenceRegionDisplaySpacing = 3;
14 opts.ShowFullContour = 'off';
15
16 figure(1)
17 nyquistplot(Gf1,G1,freqs,opts,'sd',2.45);
18 legend('Non-parametric','Parametric')
19 % % add a zoomed zone
20 % box on
21 % zp = BaseZoom();
22 % zp.run;
23
24 % logs_3
25 load logs_3.mat
26 G3 = oe(data, [8 8 1]); % Parametric
27 Gf3 = spa(data,8191,freqs); % Non-parametric
28
29 opts = nyquistoptions;
30 opts.ConfidenceRegionDisplaySpacing = 3;
31 opts.ShowFullContour = 'off';
32
33 figure(2)
```

```

34 nyquistplot(Gf3,G3,freqs,opts,'sd',2.45);
35 legend('Non-parametric','Parametric')
36
37 % logs_5
38 load logs_5.mat
39 G5 = oe(data, [8 8 1]); % Parametric
40 Gf5 = spa(data,8191,freqs); % Non-parametric
41
42 opts = nyquistoptions;
43 opts.ConfidenceRegionDisplaySpacing = 3;
44 opts.ShowFullContour = 'off';
45
46 figure(3)
47 nyquistplot(Gf5,G5,freqs,opts,'sd',2.45);
48 legend('Non-parametric','Parametric')
49
50 % logs_7
51 load logs_7.mat
52 G7 = oe(data, [8 8 1]); % Parametric
53 Gf7 = spa(data,8191,freqs); % Non-parametric
54
55 opts = nyquistoptions;
56 opts.ConfidenceRegionDisplaySpacing = 3;
57 opts.ShowFullContour = 'off';
58
59 figure(4)
60 nyquistplot(Gf7,G7,freqs,opts,'sd',2.45);
61 legend('Non-parametric','Parametric')
62
63 % logs_9
64 load logs_9.mat
65 G9 = oe(data, [8 8 1]); % Parametric
66 Gf9 = spa(data,8191,freqs); % Non-parametric
67
68 opts = nyquistoptions;
69 opts.ConfidenceRegionDisplaySpacing = 3;
70 opts.ShowFullContour = 'off';
71
72 figure(5)
73 nyquistplot(Gf9,G9,freqs,opts,'sd',2.45);
74 legend('Non-parametric','Parametric')
75
76 % logs_11
77 load logs_11.mat
78 G11 = oe(data, [8 8 1]); % Parametric
79 Gf11 = spa(data,8191,freqs); % Non-parametric
80
81 opts = nyquistoptions;
82 opts.ConfidenceRegionDisplaySpacing = 3;

```

```

83 opts.ShowFullContour = 'off';
84
85 figure(6)
86 nyquistplot(Gf11,G11,freqs,opts,'sd',2.45);
87 legend('Non-parametric','Parametric')
88
89 % 2.1.5 To choose the nominal model we gonna plot the bode magnitude
90 % of the
91 % multimodel
92
93 % Bode diagram of all models
94 figure(7)
95 bodemag(G1,G3,G5,G7,G9,G11)
96 legend('G1','G3','G5','G7','G9','G11')
97
98 Gmm = stack(1,G1,G3,G5,G7,G9,G11);
99
100 % test all the nominal (by plotting the filter)
101 [Gu1, info1] = ucover(Gmm, G1, 2);
102 [Gu3, info3] = ucover(Gmm, G3, 2);
103 [Gu5, info5] = ucover(Gmm, G5, 2);
104 [Gu7, info7] = ucover(Gmm, G7, 2);
105 [Gu9, info9] = ucover(Gmm, G9, 2);
106 [Gu11,info11] = ucover(Gmm, G11, 2);
107
108 W2_1 = info1.Wlopt;
109 W2_3 = info3.Wlopt;
110 W2_5 = info5.Wlopt;
111 W2_7 = info7.Wlopt;
112 W2_9 = info9.Wlopt;
113 W2_11 = info11.Wlopt;
114
115 figure(8) % Gnom = G1
116 hold on
117 bodemag((G3/G1)-1,'b')
118 bodemag((G5/G1)-1,'b')
119 bodemag((G7/G1)-1,'b')
120 bodemag((G9/G1)-1,'b')
121 bodemag((G11/G1)-1,'b')
122 bodemag(W2_1,'r')
123 title('Nominal model : G1')
124 legend('G3/G1-1','G5/G1-1','G7/G1-1','G9/G1-1','G11/G1-1','W2_1')
125
126 figure(9) % Gnom = G3
127 hold on
128 bodemag((G1/G3)-1,'b')
129 bodemag((G5/G3)-1,'b')
130 bodemag((G7/G3)-1,'b')

```

```

130 bodemag((G9/G3)-1,'b')
131 bodemag((G11/G3)-1,'b')
132 bodemag(W2_3,'r')
133 title('Nominal model : G3')
134
135 figure(10) % Gnom = G5
136 hold on
137 bodemag((G1/G5)-1,'b')
138 bodemag((G3/G5)-1,'b')
139 bodemag((G7/G5)-1,'b')
140 bodemag((G9/G5)-1,'b')
141 bodemag((G11/G5)-1,'b')
142 bodemag(W2_5,'r')
143 title('Nominal model : G5')
144
145 figure(11) % Gnom = G7
146 hold on
147 bodemag((G1/G7)-1,'b')
148 bodemag((G3/G7)-1,'b')
149 bodemag((G5/G7)-1,'b')
150 bodemag((G9/G7)-1,'b')
151 bodemag((G11/G7)-1,'b')
152 bodemag(W2_7,'r')
153 title('Nominal model : G7')
154
155 figure(12) % Gnom = G9
156 hold on
157 bodemag((G1/G9)-1,'b')
158 bodemag((G3/G9)-1,'b')
159 bodemag((G5/G9)-1,'b')
160 bodemag((G7/G9)-1,'b')
161 bodemag((G11/G9)-1,'b')
162 bodemag(W2_9,'r')
163 title('Nominal model : G9')
164
165 figure(13) % Gnom = G11
166 hold on
167 bodemag((G1/G11)-1,'b')
168 bodemag((G3/G11)-1,'b')
169 bodemag((G5/G11)-1,'b')
170 bodemag((G7/G11)-1,'b')
171 bodemag((G9/G11)-1,'b')
172 bodemag(W2_11,'r')
173 title('Nominal model : G11')
174
175 % All W2 in one plot
176 figure(14)
177 bodemag(W2_1,W2_3,W2_5,W2_7,W2_9,W2_11)
178 title('Comparison between different weighting filter')

```

```

179 legend('W2 (Gnom = G1)', 'W2 (Gnom = G3)', 'W2 (Gnom = G5)', ...
180        'W2 (Gnom = G7)', 'W2 (Gnom = G9)', 'W2 (Gnom =
        G11)', 'Location', 'northwest')
181
182 [Gu74, info74] = uncover(Gmm, G7, 4);
183 W2 = info74.W1;
184 Gnom = G7;
185 save('W2')
186 save('Gnom')
187 save('Gmm')
188
189 %% 2.2
190 % Load data
191 load('Gnom')
192 load('Gmm')
193 load('W2')
194
195 % Robust performance condition
196 % W1S < 1 for Gmm
197 % [W1S W2T] < sqrt(2)/2 for Gnom
198
199 % Design of W1
200 num = [1 7];
201 den = [1 0.0001];
202 W1 = tf(num,den) * 1/2;
203 W1 = c2d(W1,Ts,'zoh');
204 figure(15)
205 bodemag(W1^-1,tf(2))
206
207 % Design of K (Hinf controller)
208 Kinf = mixsyn(Gnom,W1,[],W2);
209
210 T = feedback(Gmm*Kinf,1);
211 Tnom = feedback(Gnom*Kinf,1);
212 stepinfo(Tnom) % Check settling time
213 U = feedback(Kinf,Gmm);
214 Unom = feedback(Kinf,Gnom);
215 S = feedback(1,Gmm*Kinf);
216 Snom = feedback(1,Gnom*Kinf);
217
218 % Plot
219 figure(16)
220 subplot(2,2,1)
221 step(T,Tnom)
222 title('Step response')
223 legend('Multimodel','Nominal')
224 subplot(2,2,2)
225 step(U,Unom)
226 title('Control signal')

```

```

227 legend('Multimodel','Nominal')
228 ylim([-1.5,1.5])
229 subplot(2,2,3)
230 bodemag(U,Unom)
231 xlim([10^-2 1560])
232 title('Sensitivity function U')
233 legend('Multimodel','Nominal')
234 subplot(2,2,4)
235 bodemag(S,Snom, W1^-1)
236 xlim([10^-2 1560])
237 title('Sensitivity function S')
238 legend('Multimodel','Nominal','W1 inv')
239
240 condition = norm([W1*Snom W2*Tnom],inf)
241 if (condition <= 1/sqrt(2))
242     fprintf('Robust performance conditions are met\n')
243 end
244
245 % W3 to limit the magnitude of U
246 W3 = 0.1;
247 Kinf_range = mixsyn(Gnom, W1, W3, W2);
248
249 T = feedback(Gmm*Kinf_range,1);
250 Tnom = feedback(Gnom*Kinf_range,1);
251 U =feedback(Kinf_range,Gmm);
252 Unom =feedback(Kinf_range,Gnom);
253 S = feedback(1,Gmm*Kinf_range);
254 Snom = feedback(1,Gnom*Kinf_range);
255
256 % Plot
257 figure(17)
258 subplot(2,2,1)
259 step(T,Tnom)
260 title('Step response')
261 legend('Multimodel','Nominal')
262 subplot(2,2,2)
263 step(U,Unom)
264 title('Control signal')
265 legend('Multimodel','Nominal')
266 ylim([-1.5,1.5])
267 subplot(2,2,3)
268 bodemag(U,Unom,tf(1/W3))
269 title('Sensitivity function U')
270 legend('Multimodel','Nominal','W3^-1')
271 xlim([10^-2 1560])
272 subplot(2,2,4)
273 bodemag(S,Snom, W1^-1)
274 title('Sensitivity function S')
275 xlim([10^-2 1560])

```

```

276 legend('Multimodel','Nominal', 'W1 inv')
277
278 condition = norm([W1*Snom W2*Tnom],inf)
279
280 if (condition <= 1/sqrt(2))
281     fprintf('Robust performance conditions are met\n')
282 end
283
284 % Reduce order of K
285 orderK = size(Kinf_range.A,1);
286
287 K_ft = ss2tf(Kinf_range.A,Kinf_range.B,Kinf_range.C,Kinf_range.D);
288 Kreduced = reduce(Kinf_range, 11); % 11th-order
289 orderKreduced = size(Kreduced.A,1);
290
291 % Comparison pole-zero map
292 figure(18)
293 subplot(1,2,2)
294 pzmap(Kreduced)
295 title('Controller after reduction')
296 subplot(1,2,1)
297 pzmap(Kinf_range)
298 title('Controller before reduction')
299
300 % Comparison controllers
301 figure(19)
302 bodemag(Kinf_range,Kreduced)
303 legend('Controller before reduction', 'Controller after reduction')
304 title('Reduction of the order of the controller')
305
306 % Test performance
307 T = feedback(Gmm*Kreduced,1);
308 Tnom = feedback(Gnom*Kreduced,1);
309 U =feedback(Kreduced,Gmm);
310 Unom =feedback(Kreduced,Gnom);
311 S = feedback(1,Gmm*Kreduced);
312 Snom = feedback(1,Gnom*Kreduced);
313
314 figure(20)
315 subplot(2,2,1)
316 step(T,Tnom)
317 title('Step response')
318 legend('Multimodel','Nominal')
319 subplot(2,2,2)
320 step(U,Unom)
321 title('Control signal')
322 legend('Multimodel','Nominal')
323 ylim([-1.5,1.5])
324 subplot(2,2,3)

```

```

325 bodemag(U,Unom, 1/tf(W3))
326 xlim([10^-2 1560])
327 title('Sensitivity function U')
328 legend('Multimodel','Nominal','W3^-1')
329 subplot(2,2,4)
330 bodemag(S,Snom, W1^-1)
331 xlim([10^-2 1560])
332 title('Sensitivity function S')
333 sgtitle('Model-based H∞ control design')
334 legend('Multimodel','Nominal','W1 inv')
335
336 condition1 = norm([W1*Snom W2*Tnom],inf)
337 if (condition <= 1/sqrt(2))
338     fprintf('Robust performance conditions is met\n')
339 end
340
341 % Infinity norm
342 norm_nominal = norm([W1*Snom W2*Tnom],inf)
343 norm_multimodel = norm([W1*S W2*T],inf)
344
345 % Modify report + report Datadriven
346 %% H2 controller
347 % Conversion from discrete time to continuous time
348 Gct = d2c(Gnom);
349 [A,B,C,D] = ssdata(Gct);
350
351 % Optimization problem definition
352 n = size(A,1);
353 m = size(B,2);
354
355 % Decision variables
356 L = sdpvar(n,n,'symmetric');
357 X = sdpvar(m,n);
358 M = sdpvar(m,m);
359
360 obj = trace(C*L*C') + trace(M); % Objective function
361
362 % LMI definitions
363 lmi1 = A*L - B*X + L*A' - X'*B' + B*B' <= 0;
364 lmi2 = [M X; X' L] >= 0;
365 lmi3 = L >= 0;
366 lmi = [lmi1,lmi2,lmi3];
367
368 % Options
369 options = sdpsettings('solver','mosek');
370 optimize(lmi,obj,options);
371
372 % Controller
373 X = value(X);

```



```

374 L = value(L);
375 K_H2 = X*inv(L);
376
377 % Step response of closed loop system
378 Acl = A - B*K_H2;
379 Bcl = B;
380 Ccl = C;
381 Dcl = D;
382 sys_cl = ss(Acl,Bcl,Ccl,Dcl);
383
384 figure(21)
385 step(sys_cl)
386 title('Step response using K_{H2} controller')
387
388 % Step response with LQR
389 Q = C'*C;
390 R = eye(m);
391 [K_lqr,~,~] = lqr(A,B,Q,R);
392 Acl_lqr = A - B*K_lqr;
393 sys_cl_lqr = ss(Acl_lqr,Bcl,Ccl,Dcl);
394
395 % Comparison
396 figure(22)
397 step(sys_cl_lqr, '*')
398 hold on
399 step(sys_cl, '-g')
400 legend('Step response using LQR', 'Step response using H2')
401 title('Step response comparison')
402
403 %% Data-driven controller - multimodel
404 % Load empty structure
405 [SYS, OBJ, CON, PAR] = datadriven.utils.emptyStruct();
406
407 %
408 % -----
409 % Initial controller
410 % -----
411
410 z = tf('z',Ts);
411 c = 0.001;
412 Kc = c / (1 - z^-1); % Initial controller
413 [num, den] = tfdata(Kc, 'v'); % Extract numerator and denominator
414
415 order = orderKreduced;
416 den(order + 1) = 0; % Zero padding to have same order as desired
417 % controller
418 num(order + 1) = 0; % Zero padding to have same order as desired
419 % controller
420
421

```

```

419 % Fixed parts of the controller
420 % NOTE: Initial controller should contain the fixed parts too!
421 Fy = [1 -1]; % Fixed part of denominator as
      polynomial
422 den = deconv(den, Fy); % Remove fixed part of denominator
423
424 Fx = 1; % Fixed part of numerator as polynomial
425 num = deconv(num, Fx); % Remove fixed part of numerator
426
427 SYS.controller.num = num;
428 SYS.controller.den = den;
429 SYS.controller.Ts = Ts;
430 SYS.controller.Fx = Fx;
431 SYS.controller.Fy = Fy;
432
433 %
      -----
434 % Nominal system(s)
435 %
      -----
436 % Systems should be LTI systems ('ss', 'tf', 'frd', ...)
437 SYS.model = Gmm;
438
439 %
      -----
440 % Frequencies for controller synthesis
441 %
      -----
442 SYS.W = logspace(0, log10(pi/Ts), 400);
443
444 %
      -----
445 % Filters for objectives
446 %
      -----
447 % Filter should be LTI systems ('ss', 'tf', 'frd', ...)
448 % For unused objectives, set filters to []
449 OBJ.oinf.W1 = W1; % W1 S
450 OBJ.oinf.W2 = []; % W2 T
451 OBJ.oinf.W3 = tf(W3); % W3 U
452 OBJ.oinf.W4 = []; % W4 V
453
454 OBJ.o2.W1 = []; % W1 S
455 OBJ.o2.W2 = []; % W2 T
456 OBJ.o2.W3 = []; % W3 U
457 OBJ.o2.W4 = []; % W4 V
458
459 OBJ.LSinf.Ld = []; % W (Ld - G K)
460 OBJ.LSinf.W = [];

```

```

461
462 OBJ.LS2.Ld = [];      % W (Ld - G K)
463 OBJ.LS2.W = [];
464
465 %
466 -----
466 % Filters for constraints
467 %
467 -----
468 % Filter should be LTI systems ('ss', 'tf', 'frd', ...)
469 % For unused constraints, set filters to []
470 CON.W1 = [];          % W1 S 1
471 CON.W2 = [];          % W2 T 1
472 CON.W3 = [];          % W3 U 1
473 CON.W4 = [];          % W4 V 1
474
475 %
476 -----
476 % Optimisation parameters
477 %
477 -----
478 PAR.tol = 1e-4;        % Numerical tolerance for convergence
479 PAR.maxIter = 100;     % Maximum number of allowed iterations
480
481 verbosity = true;      % To print controller synthesis iterations
482 solver = "mosek";      % Solver to use for optimisation ("mosek",
483 "sedumi", ...)
484
485 %
486 -----
485 % Solve the datadriven controller synthesis problem
486 %
486 -----
487 [K, sol] = datadriven.datadriven(SYS, OBJ, CON, PAR, verbosity,
488 solver);
489
489 % Test performance
490 T = feedback(Gmm*K, 1);
491 Tnom = feedback(Gnom*K, 1);
492 U = feedback(K, Gmm);
493 Unom = feedback(K, Gnom);
494 S = feedback(1, Gmm*K);
495 Snom = feedback(1, Gnom*K);
496
497 figure(21)
498 subplot(2,2,1)
499 step(T, Tnom)
500 title('Step response')
501 legend('Multimodel', 'Nominal')

```

```

502 subplot(2,2,2)
503 step(U,Unom)
504 title('Control signal')
505 legend('Multimodel','Nominal')
506 ylim([-1.5,1.5])
507 subplot(2,2,3)
508 bodemag(U,Unom, 1/tf(W3))
509 xlim([10^-2 1560])
510 title('Sensitivity function U')
511 legend('Multimodel','Nominal','W3^-1')
512 subplot(2,2,4)
513 bodemag(S,Snom, W1^-1)
514 xlim([10^-2 1560])
515 title('Sensitivity function S')
516 legend('Multimodel','Nominal','W1 inv')
517 sgtitle('Datadriven - Multimodel')
518 saveas(gca,'CE2_datadriven_multimodel','png')
519
520 condition = norm(W1*S,inf);
521
522 if all(condition <= 1)
523     fprintf('Robust performance conditions are met\n')
524 end
525
526 % Reduction of the controller
527 Kred = reduce(K,8);
528
529 figure(23)
530 bodemag(K,Kred)
531 legend('Controller before reduction','Controller after reduction')
532 title('Reduction of the order of the controller')
533
534 % Test performance
535 T = feedback(Gmm*Kred,1);
536 Tnom = feedback(Gnom*Kred,1);
537 U = feedback(Kred,Gmm);
538 Unom = feedback(Kred,Gnom);
539 S = feedback(1,Gmm*Kred);
540 Snom = feedback(1,Gnom*Kred);
541
542 figure(24)
543 subplot(2,2,1)
544 step(T,Tnom)
545 title('Step response')
546 legend('Multimodel','Nominal')
547 subplot(2,2,2)
548 step(U,Unom)
549 title('Control signal')
550 legend('Multimodel','Nominal')

```

```

551 ylim([-1.5,1.5])
552 subplot(2,2,3)
553 bodemag(U,Unom, 1/tf(W3))
554 xlim([10^-2 1560])
555 title('Sensitivity function U')
556 legend('Multimodel','Nominal','W3^-1')
557 subplot(2,2,4)
558 bodemag(S,Snom, W1^-1)
559 xlim([10^-2 1560])
560 title('Sensitivity function S')
561 legend('Multimodel','Nominal','W1 inv')
562 sgtitle('Datadriven - Multimodel reduced')
563 saveas(gca,'CE2_datadriven_multimodel_reduced','png')
564
565 condition = norm(W1*S,inf)
566
567 if all(condition <= 1)
568     fprintf('Robust performance conditions are met\n')
569 end
570 %% Data-driven controller - multiplicative
571 % Load empty structure
572 [SYS, OBJ, CON, PAR] = datadriven.utils.emptyStruct();
573
574 %
575 % -----
576 % Initial controller
577 % -----
578 z = tf('z',Ts);
579 c = 0.001;
580 Kc = c / (1 - z^-1); % Initial controller
581 [num, den] = tfdata(Kc, 'v'); % Extract numerator and denominator
582
583 order = orderKreduced;
584 den(order + 1) = 0; % Zero padding to have same order as desired
585 controller
586 num(order + 1) = 0; % Zero padding to have same order as desired
587 controller
588
589 % Fixed parts of the controller
590 % NOTE: Initial controller should contain the fixed parts too!
591 Fy = [1 -1]; % Fixed part of denominator as
592 polynomial
593 den = deconv(den, Fy); % Remove fixed part of denominator
594
595 Fx = 1; % Fixed part of numerator as polynomial
596 num = deconv(num, Fx); % Remove fixed part of numerator
597
598 SYS.controller.num = num;

```

```

595 SYS.controller.den = den;
596 SYS.controller.Ts = Ts;
597 SYS.controller.Fx = Fx;
598 SYS.controller.Fy = Fy;
599
600 %
601 % -----
602 %   Nominal system(s)
603 % -----
604 % Systems should be LTI systems ('ss', 'tf', 'frd', ...)
605 SYS.model = Gnom;
606 %
607 % -----
608 %   Frequencies for controller synthesis
609 % -----
610 SYS.W = logspace(0, log10(pi/Ts), 400);
611 %
612 % -----
613 %   Filters for objectives
614 % -----
615 % Filter should be LTI systems ('ss', 'tf', 'frd', ...)
616 % For unused objectives, set filters to []
617 OBJ.oinf.W1 = W1;    % W1    S
618 OBJ.oinf.W2 = W2;    % W2    T
619 OBJ.oinf.W3 = tf(W3); % W3    U
620 OBJ.oinf.W4 = [];    % W4    V
621
622 OBJ.o2.W1 = [];      % W1    S
623 OBJ.o2.W2 = [];      % W2    T
624 OBJ.o2.W3 = [];      % W3    U
625 OBJ.o2.W4 = [];      % W4    V
626
627 OBJ.LSinf.Ld = [];   % W    (Ld - G K)
628 OBJ.LSinf.W = [];
629
630 OBJ.LS2.Ld = [];     % W    (Ld - G K)
631 OBJ.LS2.W = [];
632 %
633 % -----
634 %   Filters for constraints
635 % -----
636 % Filter should be LTI systems ('ss', 'tf', 'frd', ...)

```

```

636 % For unused constraints, set filters to []
637 CON.W1 = [];      % W1      S      1
638 CON.W2 = [];      % W2      T      1
639 CON.W3 = [];      % W3      U      1
640 CON.W4 = [];      % W4      V      1
641
642 %
643 -----
643 %   Optimisation parameters
644 %
645 -----
645 PAR.tol = 1e-4;    % Numerical tolerance for convergence
646 PAR.maxIter = 100; % Maximum number of allowed iterations
647
648 verbosity = true;  % To print controller synthesis iterations
649 solver = "mosek";  % Solver to use for optimisation ("mosek",
650                   % "sedumi", ...)
651
652 %
653 -----
653 %   Solve the datadriven controller synthesis problem
654
655 -----
654 [K, sol] = datadriven.datadriven(SYS, OBJ, CON, PAR, verbosity,
655                                 solver);
656
657 % Test performance
657 T = feedback(Gmm*K, 1);
658 Tnom = feedback(Gnom*K, 1);
659 U = feedback(K, Gmm);
660 Unom = feedback(K, Gnom);
661 S = feedback(1, Gmm*K);
662 Snom = feedback(1, Gnom*K);
663
664 figure(22)
665 subplot(2,2,1)
666 step(T, Tnom)
667 title('Step response')
668 legend('Multimodel', 'Nominal')
669 subplot(2,2,2)
670 step(U, Unom)
671 title('Control signal')
672 legend('Multimodel', 'Nominal')
673 ylim([-1.5, 1.5])
674 subplot(2,2,3)
675 bodemag(U, Unom, 1/tf(W3))
676 xlim([10^-2 1560])
677 title('Sensitivity function U')
678 legend('Multimodel', 'Nominal', 'W3^-1')

```

```

679 subplot(2,2,4)
680 bodemag(S,Snom, W1^-1)
681 xlim([10^-2 1560])
682 title('Sensitivity function S')
683 legend('Multimodel','Nominal','W1 inv')
684 sgtitle('Datadriven - Multiplicative')
685 saveas(gca,'CE2_datadriven_multiplicative','png')
686
687 condition = norm([W1*Snom W2*Tnom],inf);
688
689 if condition <= 1/sqrt(2)
690     fprintf('Robust performance conditions are met\n')
691 end
692
693 % Reduction of the controller
694 Kred = reduce(K,8);
695
696 figure(23)
697 bodemag(K,Kred)
698 legend('Controller before reduction','Controller after reduction')
699 title('Reduction of the order of the controller')
700
701 % Test performance
702 T = feedback(Gmm*Kred,1);
703 Tnom = feedback(Gnom*Kred,1);
704 U = feedback(Kred,Gmm);
705 Unom = feedback(Kred,Gnom);
706 S = feedback(1,Gmm*Kred);
707 Snom = feedback(1,Gnom*Kred);
708
709 figure(24)
710 subplot(2,2,1)
711 step(T,Tnom)
712 title('Step response')
713 legend('Multimodel','Nominal')
714 subplot(2,2,2)
715 step(U,Unom)
716 title('Control signal')
717 legend('Multimodel','Nominal')
718 ylim([-1.5,1.5])
719 subplot(2,2,3)
720 bodemag(U,Unom, 1/tf(W3))
721 xlim([10^-2 1560])
722 title('Sensitivity function U')
723 legend('Multimodel','Nominal','W3^-1')
724 subplot(2,2,4)
725 bodemag(S,Snom, W1^-1)
726 xlim([10^-2 1560])
727 title('Sensitivity function S')

```



```
728 legend('Multimodel','Nominal','W1 inv')
729 sgttitle('Datadriven - Multiplicative reduced')
730 saveas(gca,'CE2_datadriven_multiplicative_reduced','png')
731
732 condition = norm(W1*S,inf);
733
734 if all(condition <= 1)
735     fprintf('Robust performance conditions are met\n')
736 end
```