



HACETTEPE UNIVERSITY - COMPUTER ENGINEERING

Advanced Robust Control MMÜ 749

Homework-5

H_∞ Controller Design

Student Name
Student ID:

Ayça KULA
202237285

Spring 2022

1 System description

The model is simple where a force $u(t)$ is acting on a mass m . So the the plant is basically described as

$$\ddot{x} = u(t)/m$$

The measurement will be either position or speed. Therefore, in order to have a measurement of speed I have to add a derivative term. For the two problems if I have an measurement of position transfer function is defined as in algorithm 1. However, for speed measurement I have defined a new plant as in 2.

```
% Define Plant
num = 1;
den = [1 1e-3 1e-6];
% Nominal plant (Transfer function with no uncertainty)
GO = tf(num,den);
```

Algorithm 1: Plant for measurement of Position

```
% Define Plant
num = 1;
den = [1 1e-3];
% Nominal plant (Transfer function with no uncertainty)
GO = tf(num,den);
```

Algorithm 2: Plant for measurement of Speed

2 H_∞ controller design with unstructured uncertainty approach

2.1 Mass uncertainty

Each uncertain parameter is bounded within some region. Therefore the mass m is defined within a known interval:

$$m_{\min} \leq m_p \leq m_{\max}$$

where m_p is an uncertain gain. We have parameter sets of the form

$$m_p = \bar{m} (1 + r_m \Delta), \quad \bar{m} \triangleq \frac{m_{\min} + m_{\max}}{2}, \quad r_m \triangleq \frac{(m_{\max} - m_{\min})/2}{\bar{m}},$$

where r_m is the relative magnitude of the gain uncertainty and \bar{m} is the average gain. Moreover, r_m could also be defined as:

$$r_m = \frac{m_{\max} - m_{\min}}{m_{\min} + m_{\max}}$$

And this equation can also be rewritten as multiplicative uncertainty:

$$G_p(s) = \underbrace{\bar{m}G_0(s)}_{G(s)} (1 + r_m \Delta), \quad |\Delta| \leq 1$$

where Δ is a real scalar and $G(s)$ is the nominal plant. Therefore, I take the Δ value as 1.

```
% Parametric uncertainty
% Upper and lower bound for mass
m_upperBar = 1.2;
m_lowerBar = 0.8;

% Relative magnitude of the gain uncertainty
rm = (m_upperBar-m_lowerBar)/(m_upperBar+m_lowerBar);

% Any stable transfer function which at each frequency is less
% than or equal to one in magnitude.
delta = 1;

% Plant with multiplicative uncertainty
Gp = G0*(1+rm*delta);
```

Algorithm 3: Parametric Uncertainty for Mass

2.2 Time constant and time delay uncertainty for actuator

$U(t)$ is a force produced by an actuator. I have selected a delay-free nominal actuator model U_0 as given in the book:

$$U_0 = \frac{2.5}{2.5s + 1}$$

If there is an unknown time constant τ and unknown actuator time delay θ as:

$$U(s) = U_0(s) \frac{e^{-\theta s}}{1 + \tau s} \text{ where } \underline{\tau} < \tau < \bar{\tau} \text{ and } \underline{\theta} < \theta < \bar{\theta}$$

I prefer to lump the uncertainty into a multiplicative uncertainty of the form

$$\Pi_I : \quad G_p(s) = G(s) (1 + w_I(s) \Delta_I(s)); \underbrace{|\Delta_I(j\omega)| \leq 1 \forall \omega}_{\|\Delta_I\|_\infty \leq 1}$$

Here $\Delta_I(s)$ is any stable transfer function which at each frequency is less than or equal to one in magnitude. Some examples of allowable $\Delta_I(s)$'s with \mathcal{H}_∞ norm less than one, $\|\Delta_I\|_\infty \leq 1$, are

$$\frac{s - z}{s + z}, \quad \frac{1}{\tau s + 1}, \quad \frac{1}{(5s + 1)^3}, \quad \frac{0.1}{s^2 + 0.1s + 1}$$

Therefore, I have chosen $\Delta_I(s)$ value as:

$$\Delta_I(s) = \frac{1}{(5s + 1)^3}$$

To derive $w_I(s)$ we first try a simple first-order weight that matches this limiting behaviour as given in "Skogestad's" book.

$$w_{I1}(s) = \frac{Ts + 0.2}{(T/2.5)s + 1}, \quad T = 4$$

And finally we use:

$$w_I(s) = \omega_{I1}(s) \frac{s^2 + 1.6s + 1}{s^2 + 1.4s + 1}$$

```
% Actuator
U0 = 2.5/(2.5*s+1);
% Simple first-order weight
T = 4;
wI1 = (T*s + 0.2)/((T/2.5)*s+1);
deltaI = 1/(5*s+1)^3;
% Multiply with correction factor to lift the gain
wI = wI1*tf([1 1.6 1],[1 1.4 1]);
Up = U0*(1+wI*deltaI);

% Plant with actuator
G = Up*Gp;
```

Algorithm 4: Parametric Uncertainty for Actuator

2.3 Obtaining generalized plant

The vector output for a servo model which is shown in figure 1, can be defined as:

$$z_1 = W_1(s)T(s)V(s)\omega$$

$$z_2 = W_2(s)S(s)V(s)\omega$$

$$z_3 = W_3(s)U(s)V(s)\omega$$

where $U(s) = K(s)S(s)$.

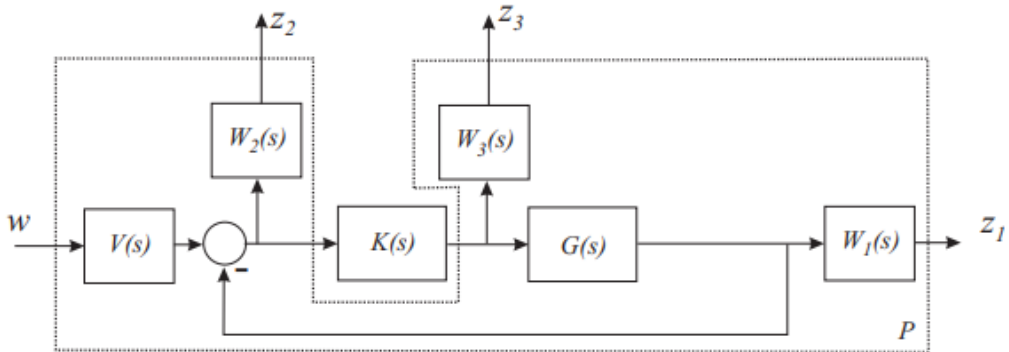


Figure 1: Servo problem

W_1 is large in frequency band where accurate tracking or disturbance rejection is required. W_2 is constant. W_3 is large in frequency band where the model is uncertain. Therefore, for both servo and regulator problems the weighting functions

are taken as (Copied from Hinf pdf):

$$W_1 = \frac{4e06}{1e06s^2 + 1400s + 1}$$

$$W_2 = 0.01$$

$$W_3 = \frac{0.05s^2 + 7e - 05s + 5e - 08}{2.5e - 05s^2 + 0.007s + 1}$$

As seen in algorithm 5, the weights obtained is implemented into the "augw" function in Matlab and therefore the generalized plant is built. While implementing the weights into Matlab be careful that W1,W2,W3 are the weights on S , $K * S$, and T . For the servo model the command is used as in 5.

```
% Selecting weighting functions
% W1 is large frequency band where accurate tracking/disturbance rejection
% is required
wn=1e-3;
num_W1=2000^2;
den_W1=[1/wn^2 2*0.7/wn 1];
W1=tf(num_W1,den_W1);

% W2 is constant
num_W2=1e-3;
den_W2=1;
W2=tf(num_W2,den_W2);

% W3 is large frequency band where model is uncertain
num_W3=5e-8*[1/wn^2 2*0.7/wn 1];
den_W3=[1/200^2 2*0.7/200 1];
W3=tf(num_W3,den_W3);

% Calculation of augmented plant:
P = augw(G,W1,W2,W3);
```

Algorithm 5: Obtaining generalized plant for servot

For a regulator model, shown in figure 2, the vector output can be defined as:

$$z_1 = W_1(s)S(s)V(s)\omega$$

$$z_2 = -W_2(s)U(s)V(s)\omega$$

$$z_3 = -W_3(s)T(s)V(s)\omega$$

where $U(s) = K(s)S(s)$.

Here you have to modify the weights in "augw" command because now the weights W1,-W2,-W3 are the weights on S , $K * S$, and T as seen in ???. The frequency response for weights can be seen in figure 3. The γ value for H_{inf} controller is 0.9350.

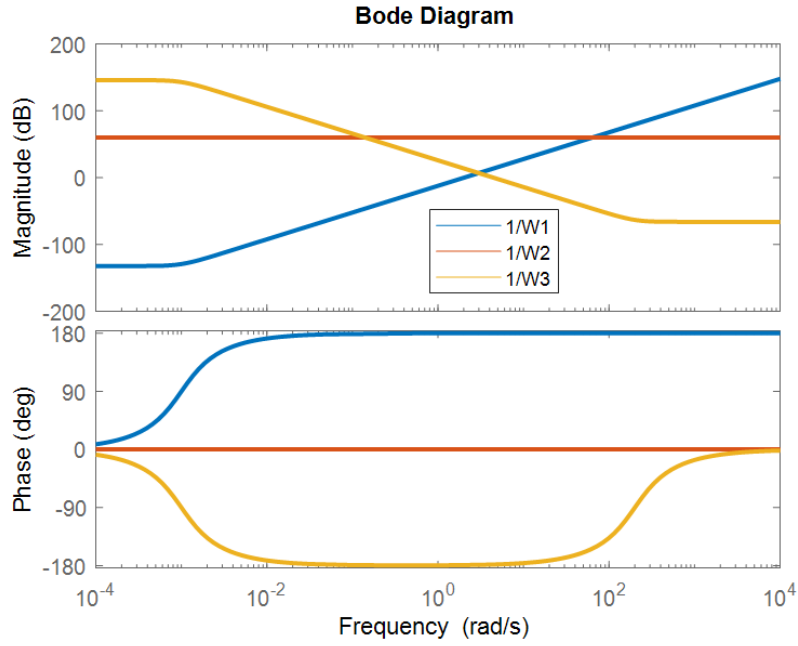


Figure 3: Frequency response for weights

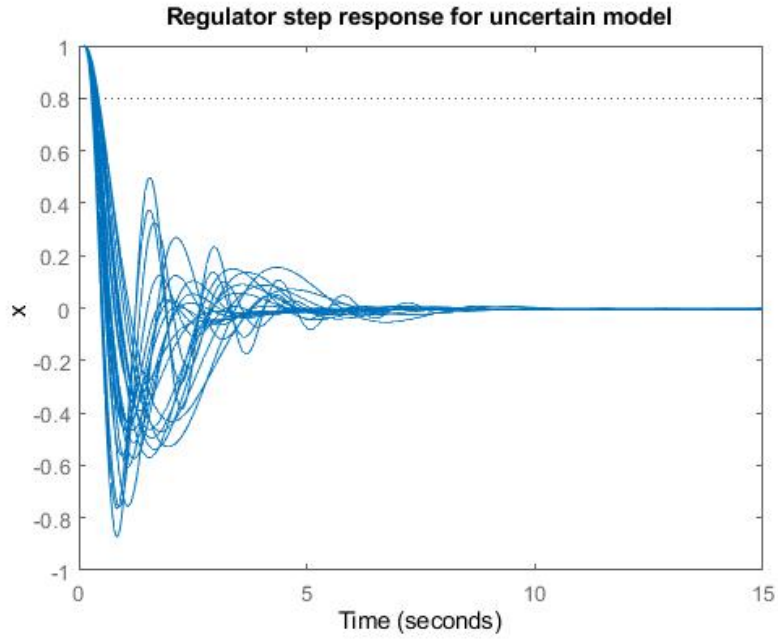


Figure 4: Step response for regulation problem

2.6 Regulation problem 2: Disturbance (sinusoidal) is acting on the plant and Measurement is speed

The figure 8 show the regulation problem where the measurement is speed. And in figure 9 the response is shown using the uncertainty. Also, you can see that the

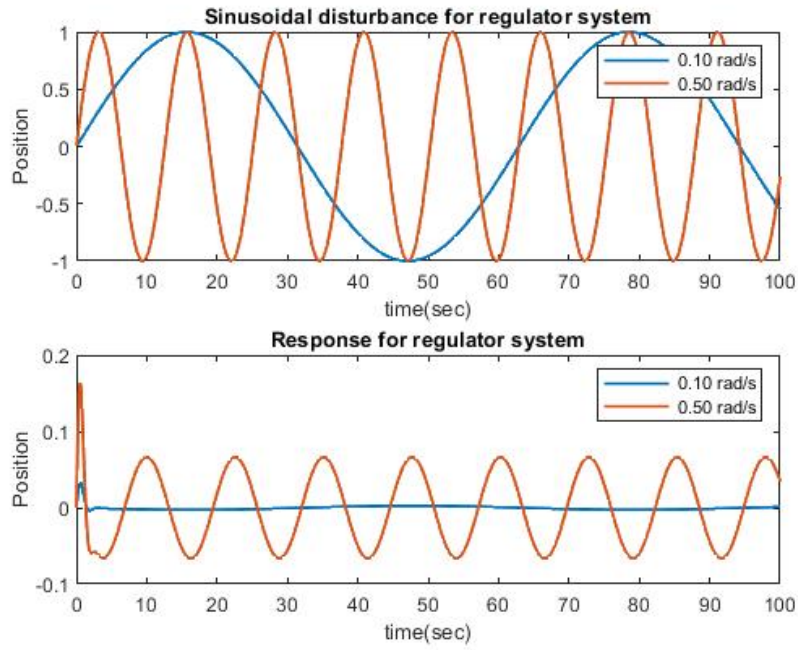


Figure 5: Regulation problem for position measurement

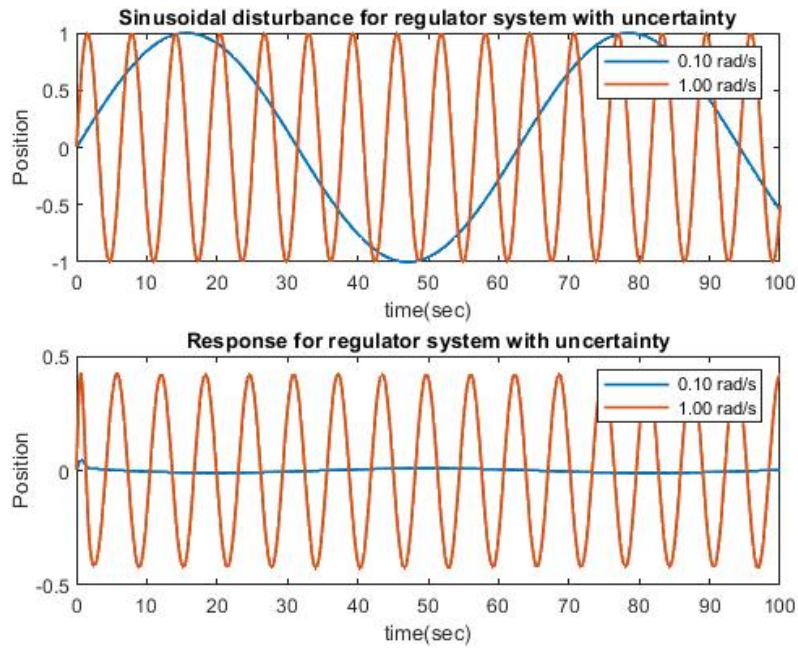


Figure 6: Regulation problem with Uncertainty for position measurement

controller can handle the disturbance.

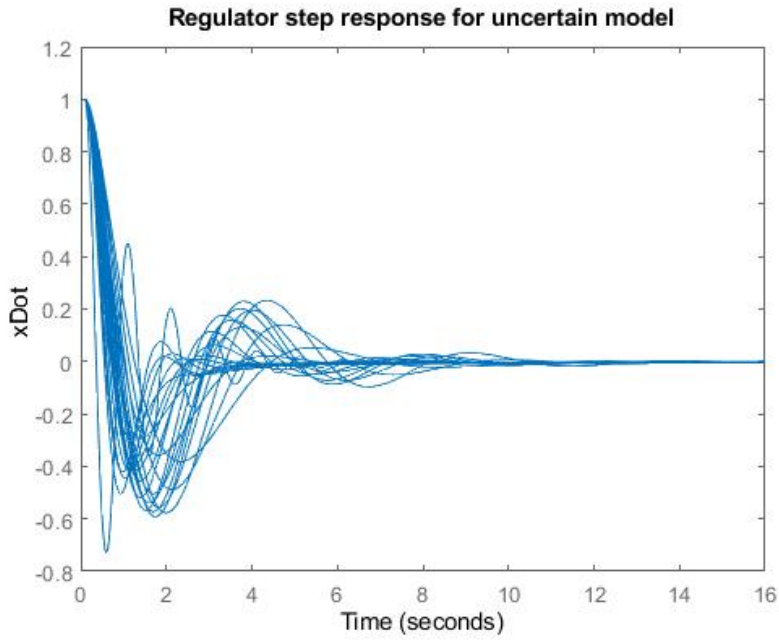


Figure 7: Step response for regulation problem

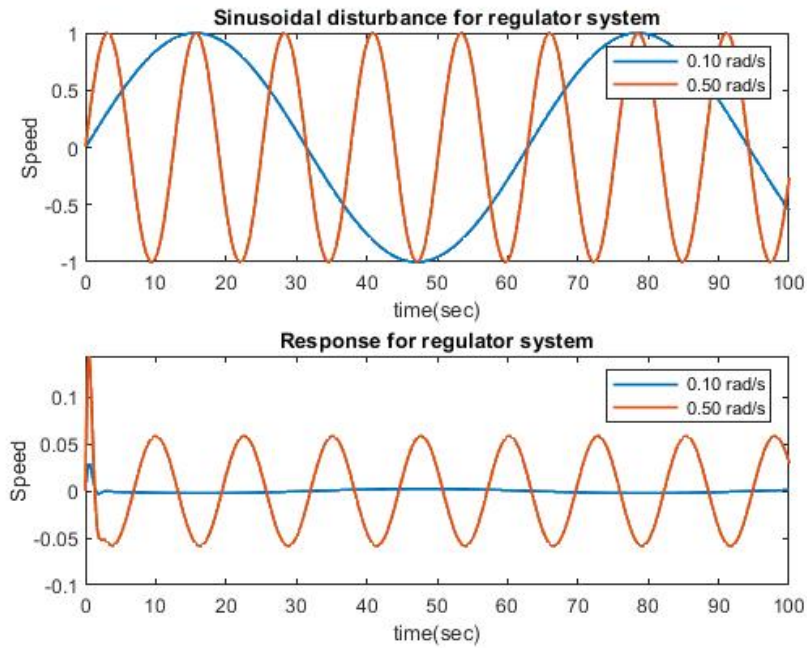


Figure 8: Regulation problem for speed measurement

2.7 Servo problem 1: Reference (sinusoidal) is position and Measurement is position

The figure 11 show the servo problem where the measurement is position. And in figure 12 the response is shown using the uncertainty. Also, you can see that the

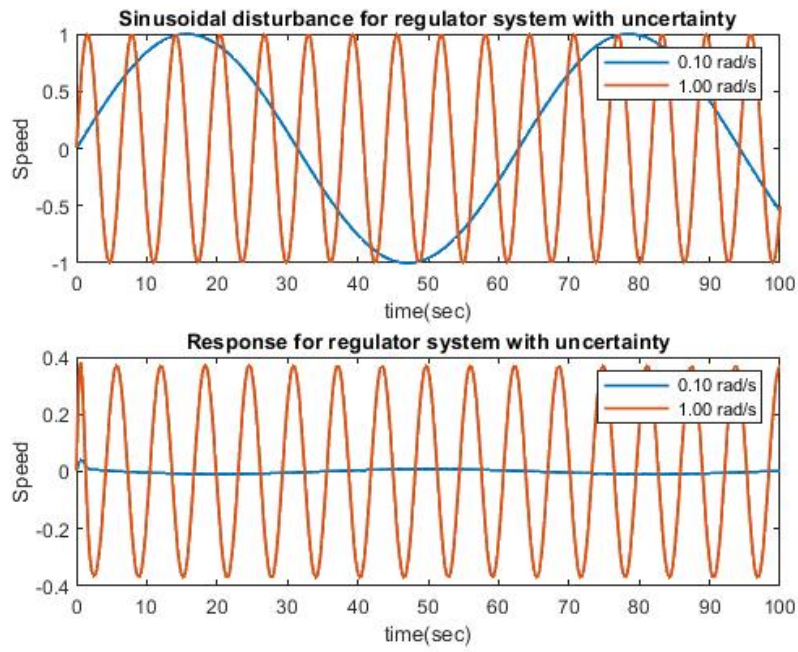


Figure 9: Regulation problem with Uncertainty for speed measurement

controller can track the reference.

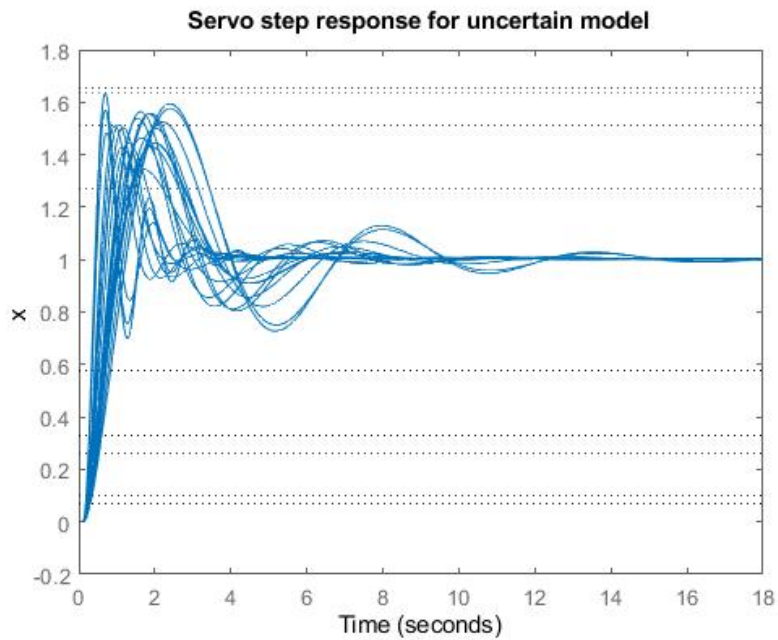


Figure 10: Step response for servo problem

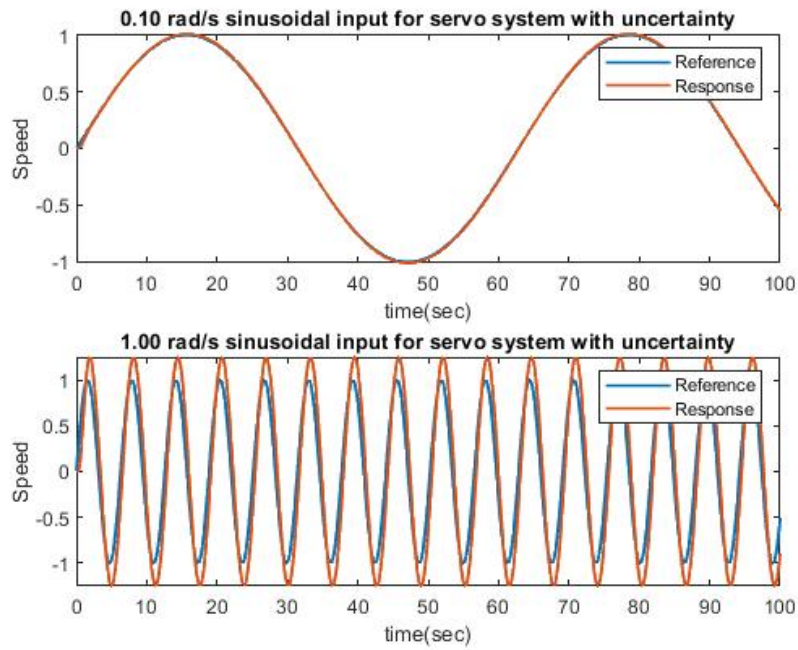


Figure 11: Servo problem for position measurement

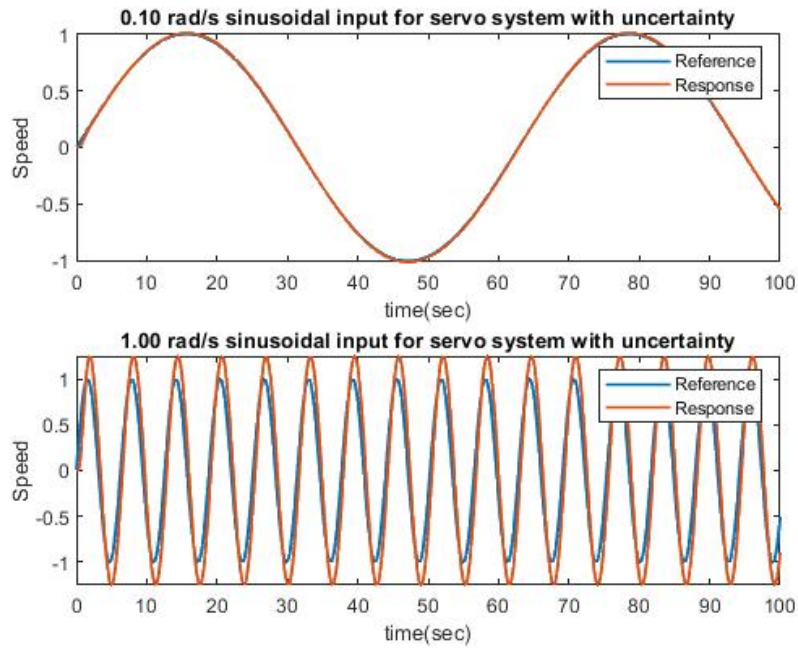


Figure 12: Servo problem with Uncertainty for position measurement

2.8 Servo problem 2: Reference (sinusoidal) is position and Measurement is speed (How?)

Figure 13 shows the response for the servo problem where the measurement is speed. The controller can track the reference.

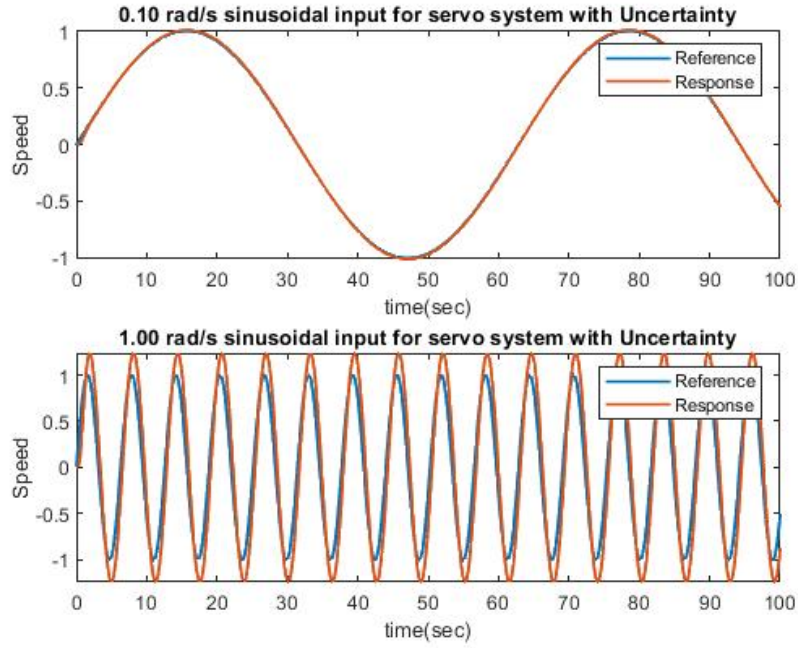


Figure 13: Servo problem with Uncertainty for speed measurement

2.9 Mixed scenarios: both reference and disturbance inputs are applied

Both reference and disturbance are added for plotting figures 14 and 15. When the control input is higher (high frequency), the controller can not control the system.

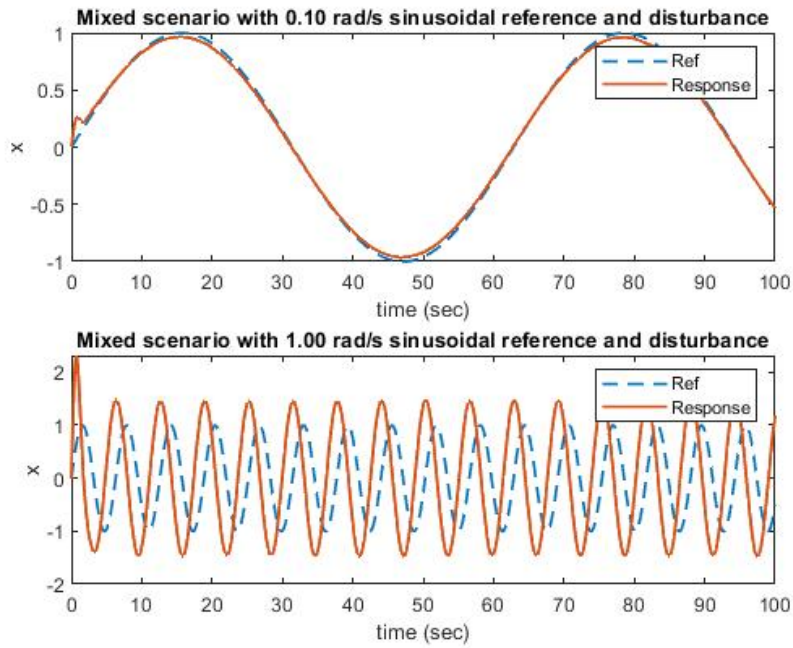


Figure 14: Mixed scenarios for position measurement

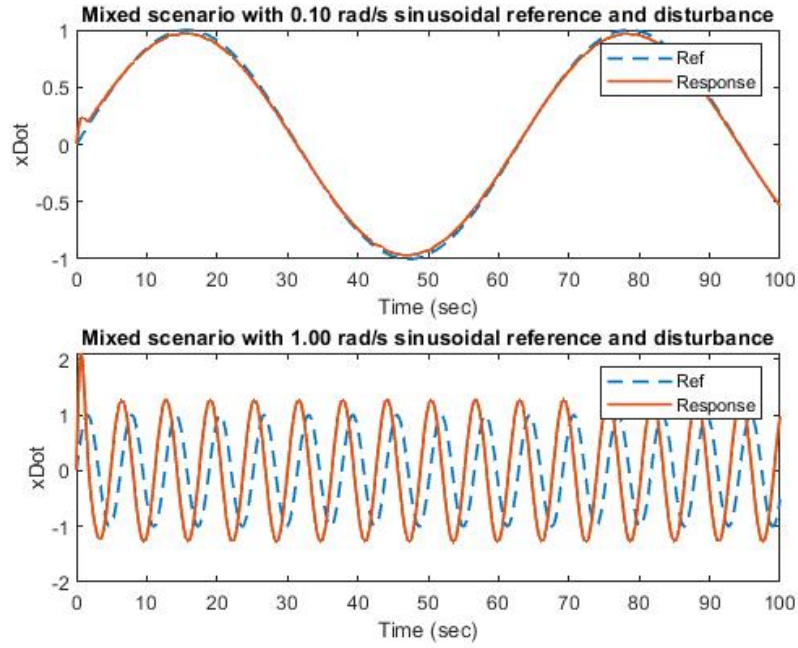


Figure 15: Mixed scenarios for speed measurement

2.10 Discussion for robust stability

Is the system stable for all different plants with different masses/ actuator time constants?

The step responses for regulator are shown in figure 7, 4 and for the servo in figure 10. Therefore, I can say that with uncertain mass and time constant H_{inf} controller can handle problems.

Is the system able to follow sinusoidal references while rejecting sinusoidal disturbances?

Yes, the system is able to follow sinusoidal references while rejecting sinusoidal disturbances as seen from previous figures.

3 Parametric uncertainty Modelling

3.1 Position Measurement

Lets model our system:

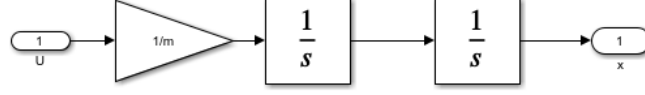


Figure 16: Simple Mass System

And now, we have to model our actuator system. However, the question says that we do not have to include the time delay uncertainty. Therefore, we define our actuator as:

$$U(s) = \frac{1}{1 + \tau s}$$

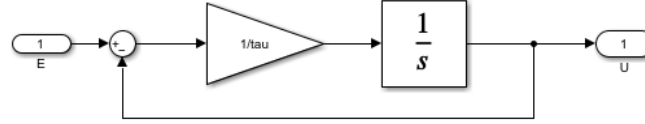


Figure 17: Actuator System

In a realistic system, the two physical parameters m and τ are not known exactly. However, it can be assumed that their values are within certain, known intervals as:

$$m = \bar{m} (1 + p_m \delta_m), \tau = \bar{\tau} (1 + p_\tau \delta_\tau)$$

where $\bar{m} = 3, \bar{\tau} = 1$ are the so-called nominal values of m and τ . p_m, p_τ, δ_m and δ_τ represent the possible (relative) perturbations on these two parameters.

We can represent a linear fractional transformation (LFT) for mass and τ respectively as:

$$M_{mi} = \begin{bmatrix} -p_m & \frac{1}{\bar{m}} \\ -p_m & \frac{1}{\bar{m}} \end{bmatrix}$$

$$M_{\tau i} = \begin{bmatrix} -p_\tau & \frac{1}{\bar{\tau}} \\ -p_\tau & \frac{1}{\bar{\tau}} \end{bmatrix}$$

All these LFTs are depicted by block diagrams in 18.

The equations relating all “inputs” to corresponding “outputs” around these perturbed parameters can now be obtained as:

$$\begin{bmatrix} y_m \\ \ddot{x} \end{bmatrix} = \begin{bmatrix} -p_m & \frac{1}{\bar{m}} \\ -p_m & \frac{1}{\bar{m}} \end{bmatrix} \begin{bmatrix} u_m \\ U \end{bmatrix}$$

$$\begin{bmatrix} y_\tau \\ \dot{u} \end{bmatrix} = \begin{bmatrix} -p_\tau & \frac{1}{\bar{\tau}} \\ -p_\tau & \frac{1}{\bar{\tau}} \end{bmatrix} \begin{bmatrix} u_\tau \\ E - U \end{bmatrix}$$

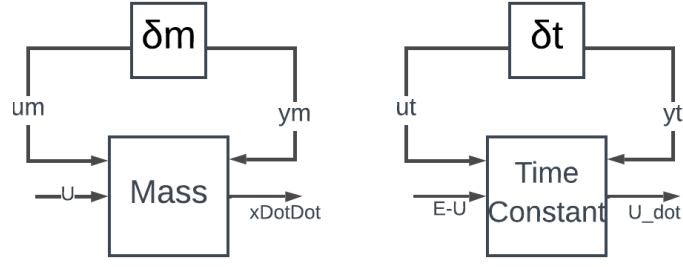


Figure 18: Representation of uncertain parameters as LFTs

$$u_m = \delta_m y_m$$

$$u_\tau = \delta_\tau y_\tau$$

Now let us set

$$x_1 = x, x_2 = \dot{x} = \dot{x}_1, y = x_1$$

such that

$$\dot{x}_2 = \ddot{x} = \ddot{x}_1$$

As a result, we obtain the following equations

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -p_m u_m + \frac{1}{\bar{m}} (U) \\ y_m &= -p_m u_m + \frac{1}{\bar{m}} (U) \\ y_\tau &= -p_\tau u_\tau + \frac{1}{\bar{\tau}} (E - U) \\ \dot{U} &= -p_\tau u_\tau + \frac{1}{\bar{\tau}} (E - U) \\ y &= x_1 \\ u_m &= \delta_m y_m \\ u_\tau &= \delta_\tau y_\tau \end{aligned}$$

The equations governing the system dynamic behaviour are given by:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{U} \\ y_m \\ y_\tau \\ y \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{\bar{m}} & -p_m & 0 & 0 \\ 0 & 0 & -\frac{1}{\bar{\tau}} & 0 & -p_\tau & \frac{1}{\bar{\tau}} \\ 0 & 0 & \frac{1}{\bar{m}} & -p_m & 0 & 0 \\ 0 & 0 & -\frac{1}{\bar{\tau}} & 0 & -p_\tau & \frac{1}{\bar{\tau}} \\ 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ U \\ u_m \\ u_\tau \\ E \end{bmatrix}$$

$$\begin{bmatrix} u_m \\ u_\tau \end{bmatrix} = \begin{bmatrix} \delta_m & 0 \\ 0 & \delta_\tau \end{bmatrix} \begin{bmatrix} y_m \\ y_\tau \end{bmatrix}$$

The state space representation of G_{mds} is

$$G_{mds} = \left[\begin{array}{c|cc} A & B_1 & B_2 \\ \hline C_1 & D_{11} & D_{12} \\ C_2 & D_{21} & D_{22} \end{array} \right]$$

where

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & \frac{1}{m} \\ 0 & 0 & -\frac{1}{\tau} \end{bmatrix}, B_1 = \begin{bmatrix} 0 & 0 \\ -p_m & 0 \\ 0 & -p_\tau \end{bmatrix}, B_2 = \begin{bmatrix} 0 \\ 0 \\ \frac{1}{\tau} \end{bmatrix}$$

$$C_1 = \begin{bmatrix} 0 & 0 & \frac{1}{m} \\ 0 & 0 & -\frac{1}{\tau} \end{bmatrix}, D_{11} = \begin{bmatrix} -p_m & 0 \\ 0 & -p_\tau \end{bmatrix}, D_{12} = \begin{bmatrix} 0 \\ \frac{1}{\tau} \end{bmatrix}$$

$$C_2 = [1 \ 0 \ 0], D_{21} = [0 \ 0], D_{22} = 0$$

In order to compute the system matrix G_{mds} , I have used the code given in 9.

```
%% System Model
% Mass
m = 3;
% Time constant
tau = 1;
% Perturbation for mass
pm = 0.4;
% Perturbation for tau
ptau = 0.3;
%% State-space representation for G_mds
A = [ 0 1 0; 0 0 1/m; 0 0 -1/tau];
B1 = [ 0 0; -pm 0; 0 -ptau];
B2 = [ 0; 0; 1/tau];
C1 = [0 0 1/m; 0 0 -1/tau];
C2 = [ 1 0 0];
D11 = [-pm 0; 0 -ptau];
D12 = [0; 1/tau];
D21 = [0 0];
D22 = 0;

G = pck(A,[B1,B2],[C1;C2],[D11 D12;D21 D22]);

% This part is implemented to not see a error
% Unpack G and see the eigenvalues
[a,b,c,d] = unpck(G);
eig(a)
% We do not want zero valued eigenvalue
% Zero eigenvalue leads to error
% Therefore, we add a very small number to change the zero eigenvalue
% situation
a(2,1) = 0.0001;
eig(a)
G = pck(a,b,c,d);
```

Algorithm 8: Compute system matrix

3.1.1 Frequency Analysis of Uncertain System

The frequency responses of the perturbed open-loop system may be computed by using the command "starp" at a few different values of the perturbation parameters δ_m, δ_τ . In figure 19, two values of each perturbation are chosen, the corresponding open-loop transfer function matrices generated and frequency responses calculated and plotted.

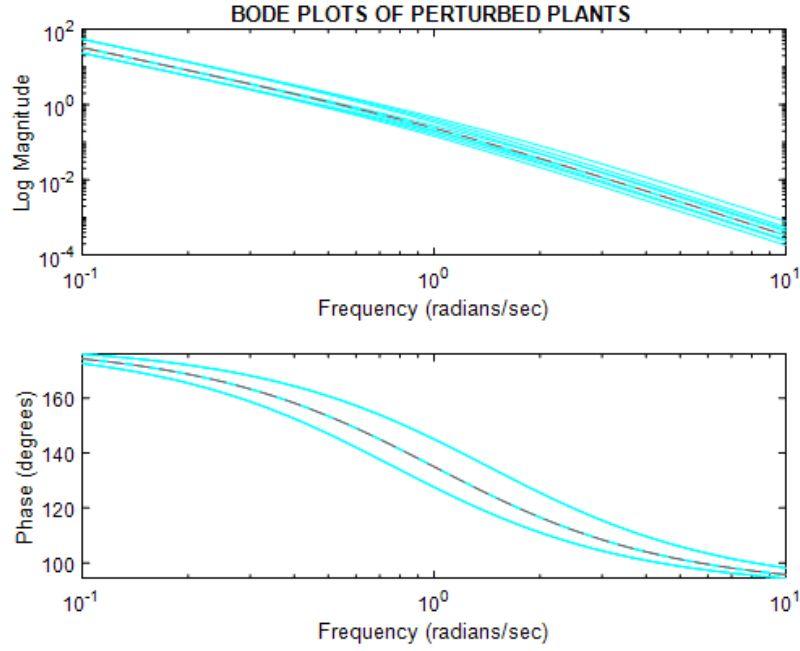


Figure 19: Frequency responses of the perturbed plants

```

%% Frequency responses of the perturbed plants
omega = logspace(-1,1,100);
[delta1,delta2] = ndgrid([-1 0 1],[-1 0 1]);
for j = 1:9
    delta = diag([delta1(j),delta2(j)]);
    olp = starp(delta,G);
    olp_ic = sel(olp,1,1);
    olp_g = frsp(olp_ic,omega);
    figure(1)
    vplot('bode',olp_g,'c-')
    subplot(2,1,1)
    hold on
    subplot(2,1,2)
    hold on
end
subplot(2,1,1)
olp_ic = sel(G,3,3);
olp_g = frsp(olp_ic,omega);
vplot('bode',olp_g,'r--')
subplot(2,1,1)
title('BODE PLOTS OF PERTURBED PLANTS')
hold off
subplot(2,1,2)
hold off

```

Algorithm 9: Frequency responses of the perturbed plants

3.1.2 Design Requirements of Closed-loop System

In the given case, the performance weighting function is a scalar function $W_p(s) = \omega_p(s)$ and chosen as:

$$W_p(s) = 0.95 \frac{s^2 + 1.8s + 10}{s^2 + 8 + 0.01}$$

which ensures, apart from good disturbance attenuation, good transient response. The control weighting function W_u is chosen as 50^{-3} . The singular values of $1/\omega_p$ over frequency range $[10^{-4}, 10^4]$. To define the weighting functions and to calculate the inverse weighting function in Matlab, the code 10 is used. It is seen in figure 20 that

from the frequency 1 rad/s the disturbance is no longer to be “attenuated”.

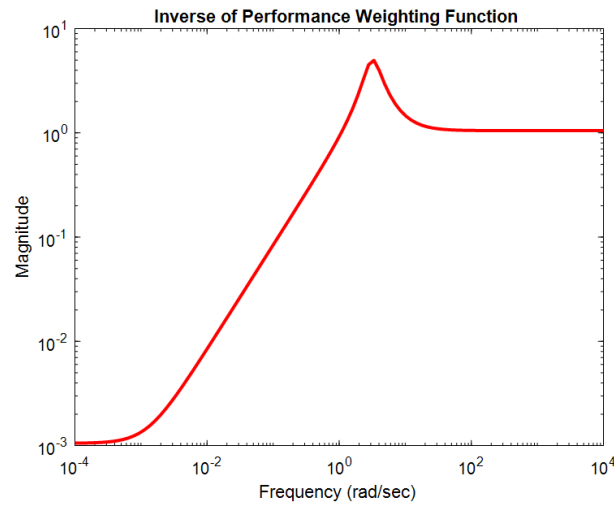


Figure 20: Singular values of $\frac{1}{\omega_p(s)}$

```
%% Design Requirements of Closed-loop System
% Robust performance:
% Define the chosen weighting functions
nuWp = [1 1.8 10];
dnWp = [1 8 0.01];
gainWp = 0.95;
Wp = nd2sys(nuWp,dnWp,gainWp);
nuWu = 1;
dnWu = 1;
gainWu = 50^(-3);
Wu = nd2sys(nuWu,dnWu,gainWu);
% Calculate the inverse weighting function
omega = logspace(-4,4,100);
Wp_g = frsp(Wp,omega);
Wpi_g = minv(Wp_g);
figure
vplot('liv,lm',Wpi_g)
title('Inverse of Performance Weighting Function')
xlabel('Frequency (rad/sec)')
ylabel('Magnitude')
```

Algorithm 10: Define weighting functions and calculate the inverse weighting function

3.1.3 System Interconnections

The variables `pertin` and `pertout` have two elements as seen in 11. The command `sysic` can be used to create the structure of open-loop systems in Matlab.

```

%% System Interconnections
systemnames = ' G Wp Wu ';
inputvar = '[ pert{2}; dist; control ]';
outputvar = '[ G(1:2); Wp; -Wu; -G(3)-dist ]';
input_to_G = '[ pert; control ]';
input_to_Wp = '[ G(3)+dist ]';
input_to_Wu = '[ control ]';
sysoutname = 'sys_ic';
cleanupsysic = 'no';
% create the structure of open-loop systems
sysic

% To analyse the open-loop system, the following commands can be used.
minfo(sys_ic)
spoles(sys_ic)
spoles(Wp)

% The model of the open-loop system with uncertainties is set
systemnames = ' G ';
inputvar = '[ pert{2}; ref; dist; control ]';
outputvar = '[ G(1:2); G(3)+dist; ref - G(3) - dist ]';
input_to_G = '[ pert; control ]';
sysoutname = 'sim_ic';
cleanupsysic = 'yes';
sysic

```

Algorithm 11: Define system interconnections

3.1.4 Suboptimal H_∞ Controller Design

This controller minimizes the infinite-norm of $F_L(P, K)$ over all stabilising controllers K . We first extract from "sys_ic" the corresponding transfer function matrix using the "sel" command as seen in 12.

```

%% 8.5 Suboptimal Hinf Controller Design
% number of measurements
nmeas = 1;
% number of controls
ncon = 1;
% lower bound of bisection
gmin = 1;
% upper bound of bisection
gmax = 10;
% absolute tolerance for the bisection method
tol = 0.001;
% open-loop interconnection is saved in the variable "hin_ic"
hin_ic = sel(sys_ic,3:5,3:4);
% K_hin: controller (matrix of type SYSTEM)
% clp: closed-loop system (matrix of type SYSTEM)
[K_hin,clp] = hinfsyn(hin_ic,nmeas,ncon,gmin,gmax,tol);

```

Algorithm 12: Compute a suboptimal H_∞ controller

The results can be seen in figure 21.

3.1.5 Analysis of Closed-loop System with K_{hin}

Figure 22 shows the singular values of the closed-loop system clp.

It is seen in figure 23 that in the low-frequency range the sensitivity function lies below $\frac{1}{\omega_p}$.

The frequency responses of the upper and lower bounds of μ are shown in Figure 24. It is clear from the figure that the closed-loop system with K_{hin} achieves robust stability.

The transient responses to the reference input and to the disturbance input are shown in 25 and 26. The transient responses are relatively slow and have slight overshoots.

```

Test bounds:      1.0000 < gamma <=      10.0000

gamma  hamx_eig  xinf_eig  hamy_eig  yinf_eig  nrho_xy  p/f
10.000  7.2e-01  4.5e-04  1.3e-03  -1.7e-16  0.0001  p
 5.500  7.2e-01  4.5e-04  1.3e-03  -6.9e-16  0.0003  p
 3.250  7.1e-01  4.5e-04  1.3e-03  -2.8e-15  0.0010  p
 2.125  7.1e-01  4.6e-04  1.3e-03  -4.9e-50  0.0025  p
 1.562  7.1e-01  4.7e-04  1.3e-03  -1.3e-23  0.0052  p
 1.281  7.0e-01  4.8e-04  1.3e-03  -1.7e-52  0.0091  p
 1.141  7.0e-01  4.9e-04  1.3e-03  -3.1e-50  0.0135  p
 1.070  6.9e-01  5.0e-04  1.3e-03  -1.7e-50  0.0174  p
 1.035  6.9e-01  5.1e-04  1.3e-03  -1.2e-51  0.0202  p
 1.018  6.9e-01  5.1e-04  1.3e-03  0.0e+00  0.0220  p
 1.009  6.9e-01  5.2e-04  1.3e-03  0.0e+00  0.0230  p
 1.004  6.9e-01  5.2e-04  1.3e-03  -6.3e-50  0.0236  p
 1.002  6.9e-01  5.2e-04  1.3e-03  -2.8e-50  0.0239  p
 1.001  6.9e-01  5.2e-04  1.3e-03  -1.7e-49  0.0240  p
 1.001  6.9e-01  5.2e-04  1.3e-03  -1.0e-51  0.0241  p

Gamma value achieved:      1.0005

```

Figure 21: Display of results for suboptimal H_∞ controller

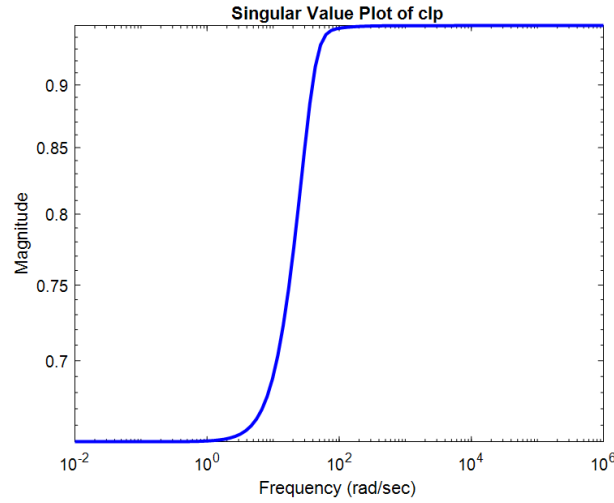


Figure 22: Singular values of the closed-loop system with K_{hin}

In order to plot the previous figures in this section use algorithm 13.

3.1.6 Regulation Problem

From figure 27, it can be seen that the controller can handle the disturbance. However, the unstructured uncertainty approach can handle the disturbance in a more better way.

3.1.7 Servo Problem

For the servo problem, this method allows the tracking to be more accurate. Therefore, the tracking is better in parametric uncertainty approach as seen in figure 28.

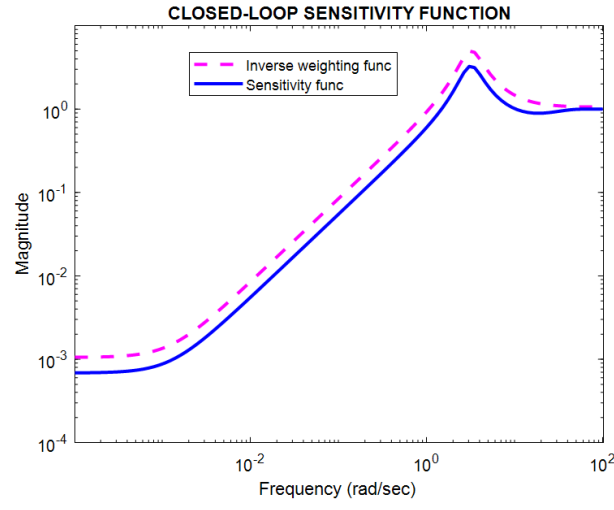


Figure 23: Sensitivity function with K_{hin}

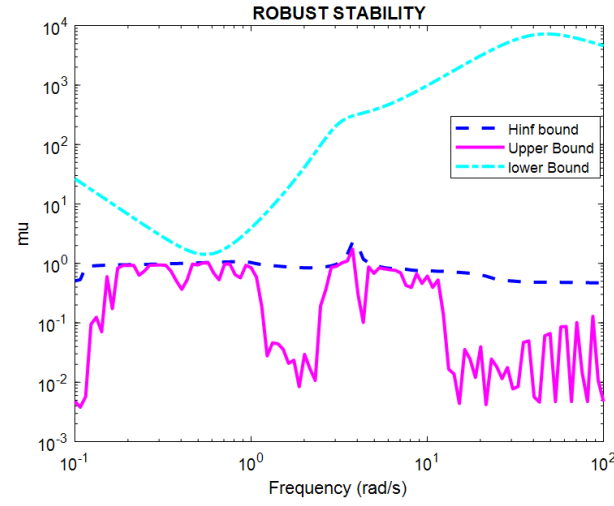


Figure 24: Robust stability analysis of K_{hin}

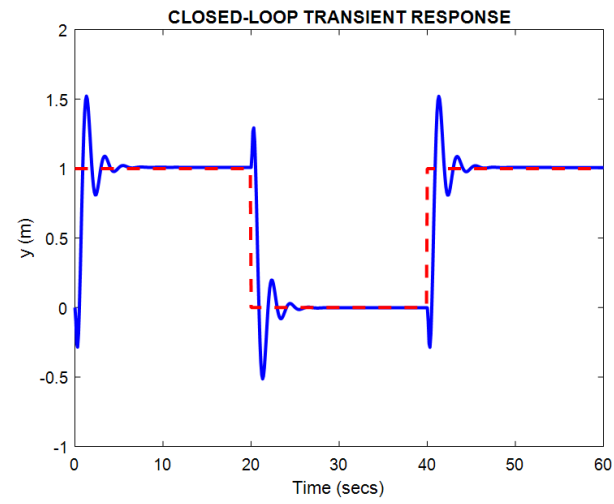


Figure 25: Transient response to reference input (K_{hin})

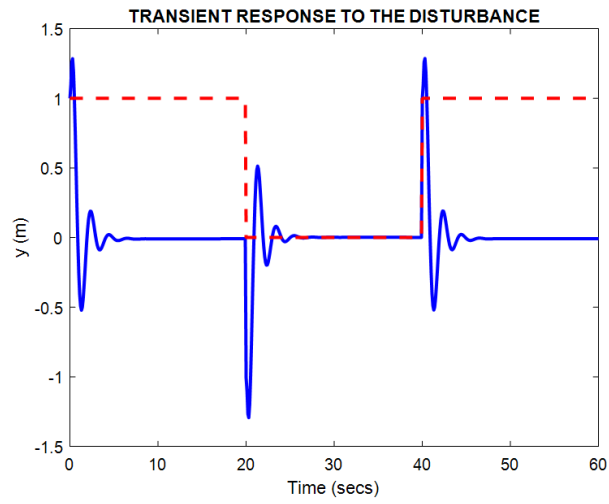


Figure 26: Transient response to disturbance input (K_{hin})

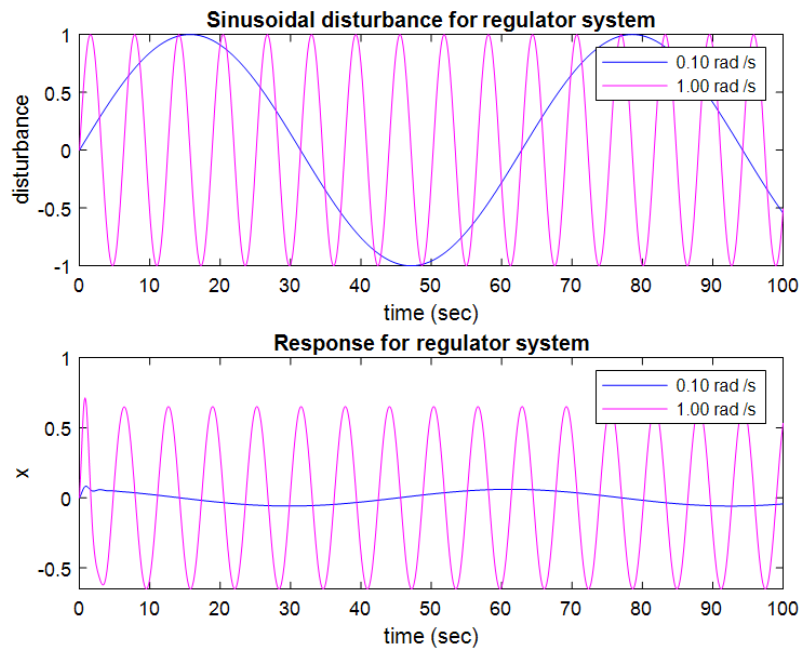


Figure 27: Sinusoidal disturbance for Regulator system for position measurement

```

% Analysis of Closed-loop System with Khin
% Singular values of the closed-loop system with Khin
minfo(K_hin)
spoles(K_hin)
omega = logspace(-2,6,100);
clp_g = frsp(clp,omega);
vplot('liv,lm',vsvd(clp_g))
title('Singular Value Plot of clp')
xlabel('Frequency (rad/sec)')
ylabel('Magnitude')

% Sensitivity function with Khin plot
K = K_hin;
clp = starp(sim_ic,K);

% inverse performance weighting function
omega = logspace(-4,2,100);
Wp_g = frsp(Wp,omega);
Wpi_g = minv(Wp_g);

% sensitivity function
sen_loop = sel(clp,3,4);
sen_g = frsp(sen_loop,omega);
vplot('liv,lm',Wpi_g,'m--',vnorm(sen_g),'y-')
title('CLOSED-LOOP SENSITIVITY FUNCTION')
xlabel('Frequency (rad/sec)')
ylabel('Magnitude')

% Robust stability analysis of Khin
clp_ic = starp(sys_ic,K);
omega = logspace(-1,2,100);
clp_g = frsp(clp_ic,omega);
blkrsR = [-1 1;-1 1;-1 1];
rob_stab = sel(clp_g,[1:3],[1:3]);
pdim = ynum(rob_stab);
fixl = [eye(pdim); 0.1*eye(pdim)]; % 1% Complex
fixr = fixl';
blkrs = [blkrsR; abs(blkrsR)];
clp_mix = mmult(fixl,rob_stab,fixr);
[rnds,rowd,sens,rowp,rowg] = mu(clp_mix,blkrs);
disp(' ')
disp(['mu-robust stability: ' ...
num2str(pkvnorm(sel(rnds,1,1)))])
disp(' ')
vplot('liv,lm',sel(rnds,1,1),'y--',sel(rnds,1,2),'m-', ...
vnorm(rob_stab),'c-.')
title('ROBUST STABILITY')
xlabel('Frequency (rad/s)')
ylabel('mu')

% Nominal and robust performance of Khin
clp_ic = starp(sys_ic,K);
omega = logspace(-1,2,100);
clp_g = frsp(clp_ic,omega);
blkrsR = [-1 1;-1 1];

% Nominal performance
nom_perf = sel(clp_g,3,3);

% Robust performance
rob_perf = clp_g;
blkrp = [blkrsR; 1 2];
bndsrp = mu(rob_perf,blkrp);
vplot('liv,lm',vnorm(nom_perf),'y-',sel(bndsrp,1,1),'m--',...
sel(bndsrp,1,2),'c-.')
tmp1 = 'NOMINAL PERFORMANCE (solid) and';
tmp2 = 'ROBUST PERFORMANCE (dashed)';
title([tmp1 tmp2])
xlabel('Frequency (rad/s)')
disp(' ')
disp(['mu-robust performance: ' ...
num2str(pkvnorm(sel(bndsrp,1,1)))])
disp(' ')

% Transient response to reference/disturbance input
clp = starp(sim_ic,K_hin);
timedata = [0 20 40];
stepdata = [1 0 1];
dist = 0;
ref = step_tr(timedata,stepdata,0.1,60);
u = abv(0,0,ref,dist);
y = trsp(clp,u,60,0.1);
figure
vplot(sel(y,3,1),'y-',ref,'r--')
title('CLOSED-LOOP TRANSIENT RESPONSE')
xlabel('Time (secs)')
ylabel('y (m)')

% Response to the disturbance
timedata = [0 20 40];
stepdata = [1 0 1];
dist = step_tr(timedata,stepdata,0.1,60);
ref = 0;
u = abv(0,0,ref,dist);
y = trsp(clp,u,60,0.1);
figure
vplot(sel(y,3,1),'y-',dist,'r--')
title('TRANSIENT RESPONSE TO THE DISTURBANCE')
xlabel('Time (secs)')
ylabel('y (m)')

```

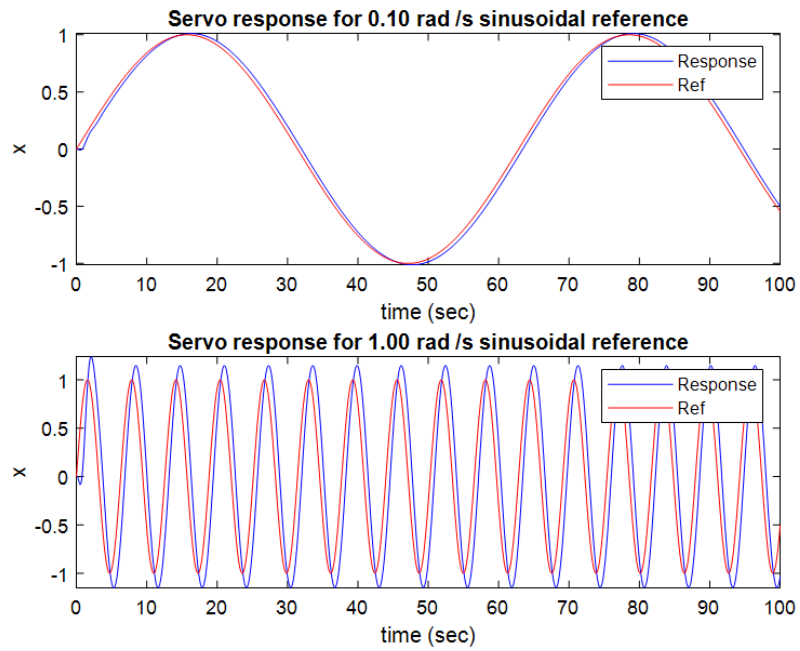


Figure 28: Servo response for different disturbances for position measurement

3.2 Speed Measurement

In order to use speed as measurement, we need to design the system as $1/s$. In this part, we only change the system matrix and we do the same calculations.

As a result, we obtain the following equations

$$\begin{aligned}\dot{x} &= -p_m u_m + \frac{1}{\bar{m}} (U) \\ y_m &= -p_m u_m + \frac{1}{\bar{m}} (U) \\ y_\tau &= -p_\tau u_\tau + \frac{1}{\bar{\tau}} (E - U) \\ \dot{u} &= -p_\tau u_\tau + \frac{1}{\bar{\tau}} (E - U) \\ y &= x \\ u_m &= \delta_m y_m \\ u_\tau &= \delta_\tau y_\tau\end{aligned}$$

The equations governing the system dynamic behaviour are given by:

$$\begin{bmatrix} \dot{x} \\ \dot{u} \\ y_m \\ y_\tau \\ y \end{bmatrix} = \begin{bmatrix} 0 & \frac{1}{\bar{m}} & -p_m & 0 & 0 \\ 0 & -\frac{1}{\bar{\tau}} & 0 & -p_\tau & \frac{1}{\bar{\tau}} \\ 0 & \frac{1}{\bar{m}} & -p_m & 0 & 0 \\ 0 & -\frac{1}{\bar{\tau}} & 0 & -p_\tau & \frac{1}{\bar{\tau}} \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ U \\ u_m \\ u_\tau \\ E \end{bmatrix}$$

$$\begin{bmatrix} u_m \\ u_\tau \end{bmatrix} = \begin{bmatrix} \delta_m & 0 \\ 0 & \delta_\tau \end{bmatrix} \begin{bmatrix} y_m \\ y_\tau \end{bmatrix}$$

The state space representation of G_{mds} is

$$G_{mds} = \left[\begin{array}{c|cc} A & B_1 & B_2 \\ \hline C_1 & D_{11} & D_{12} \\ C_2 & D_{21} & D_{22} \end{array} \right]$$

where

$$\begin{aligned}A &= \begin{bmatrix} 0 & \frac{1}{\bar{m}} \\ 0 & -\frac{1}{\bar{\tau}} \end{bmatrix}, B_1 = \begin{bmatrix} -p_m & 0 \\ 0 & -p_\tau \end{bmatrix}, B_2 = \begin{bmatrix} 0 \\ \frac{1}{\bar{\tau}} \end{bmatrix} \\ C_1 &= \begin{bmatrix} 0 & \frac{1}{\bar{m}} \\ 0 & -\frac{1}{\bar{\tau}} \end{bmatrix}, D_{11} = \begin{bmatrix} -p_m & 0 \\ 0 & -p_\tau \end{bmatrix}, D_{12} = \begin{bmatrix} 0 \\ \frac{1}{\bar{\tau}} \end{bmatrix} \\ C_2 &= \begin{bmatrix} 1 & 0 \end{bmatrix}, D_{21} = \begin{bmatrix} 0 & 0 \end{bmatrix}, D_{22} = 0\end{aligned}$$

3.2.1 Frequency Analysis of Uncertain System

The frequency responses of the perturbed open-loop system may be computed by using the command "starp" at a few different values of the perturbation parameters δ_m, δ_τ . In figure 29, two values of each perturbation are chosen, the corresponding open-loop transfer function matrices generated and frequency responses calculated and plotted.

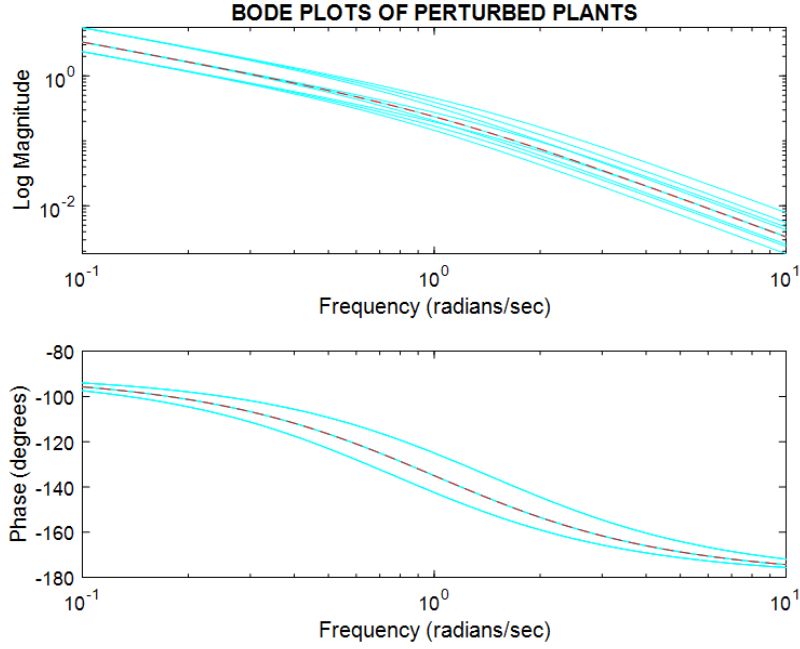


Figure 29: Frequency responses of the perturbed plants

3.2.2 Design Requirements of Closed-loop System

In the given case, the performance weighting function is a scalar function $W_p(s) = \omega_p(s)$ and chosen as:

$$W_p(s) = 0.95 \frac{s^2 + 1.8s + 10}{s^2 + 8 + 0.01}$$

which ensures, apart from good disturbance attenuation, good transient response. The control weighting function W_u is chosen as 50^{-3} . The singular values of $1/\omega_p$ over frequency range $[10^{-4}, 10^4]$. It is seen in figure 30 that from the frequency 1 rad/s the disturbance is no longer to be “attenuated”.

3.2.3 System Interconnections

The command sysic can be used to create the structure of open-loop systems in Matlab.

3.2.4 Suboptimal H_∞ Controller Design

This controller minimizes the infinite-norm of $F_L(P, K)$ over all stabilising controllers K . We first extract from "sys_ic" the corresponding transfer function matrix using the "sel" command.

3.2.5 Analysis of Closed-loop System with K_{hin}

Figure 31 shows the singular values of the closed-loop system clp.

It is seen in figure 32 that in the low-frequency range the sensitivity function lies below $\frac{1}{\omega_p}$.

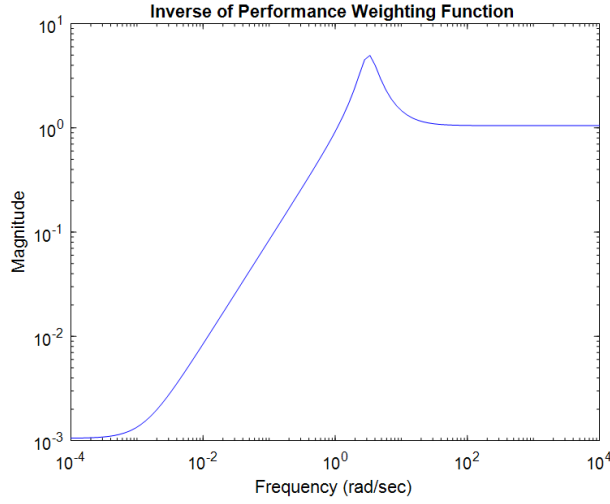


Figure 30: Singular values of $\frac{1}{\omega_p(s)}$

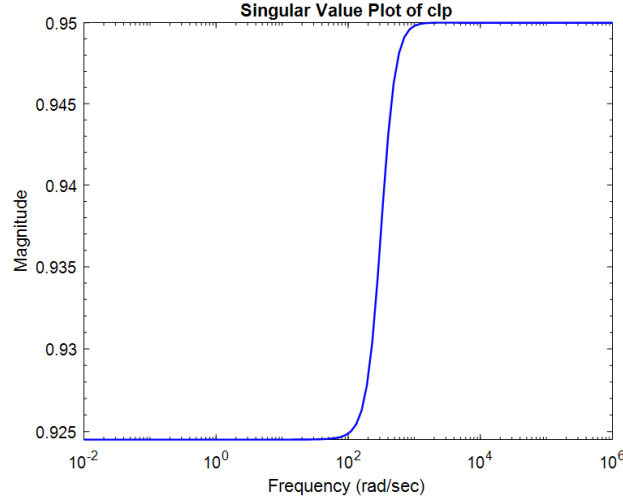


Figure 31: Singular values of the closed-loop system with K_{hin}

The frequency responses of the upper and lower bounds of μ are shown in Figure 34. It is clear from the figure that the closed-loop system with K_{hin} achieves robust stability.

The transient responses to the reference input and to the disturbance input are shown in 35 and 36. The transient responses are relatively slow and have slight overshoots.

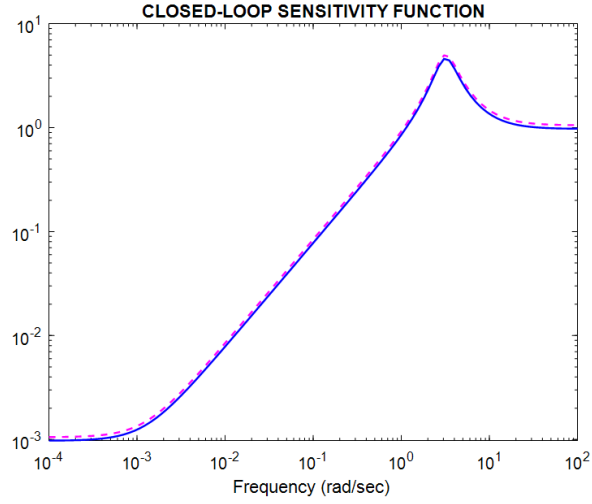


Figure 32: Sensitivity function with K_{hin}

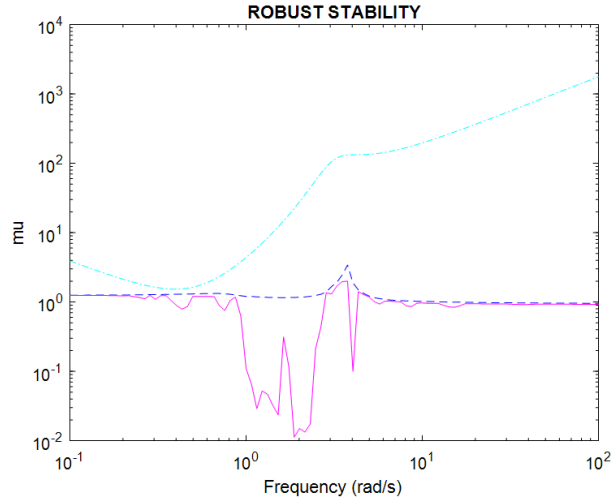


Figure 33: Robust stability analysis of K_{hin}

3.2.6 Regulation Problem

From figure 37, it can be seen that the controller can handle the disturbance. However, the unstructured uncertainty approach can handle the disturbance in a more better way.

3.2.7 Servo Problem

For the servo problem, this method allows the tracking to be more accurate. Therefore, the tracking is better in parametric uncertainty approach as seen in figure 38.

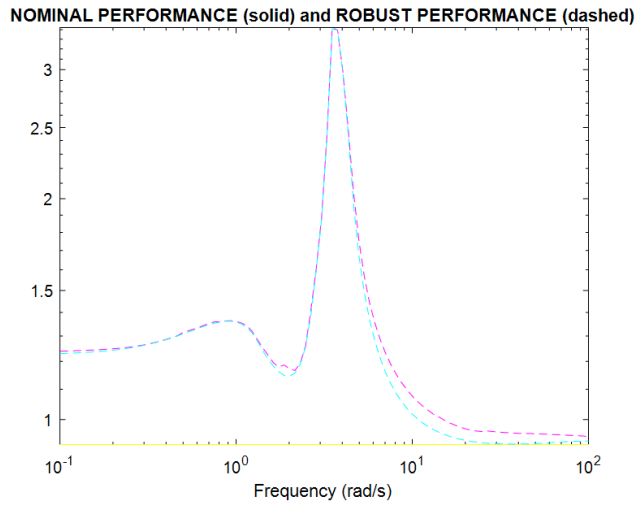


Figure 34: Nominal and robust performance

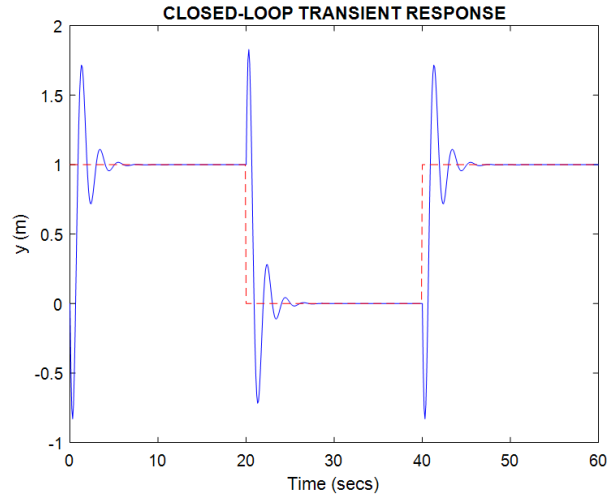


Figure 35: Transient response to reference input (K_{hin})

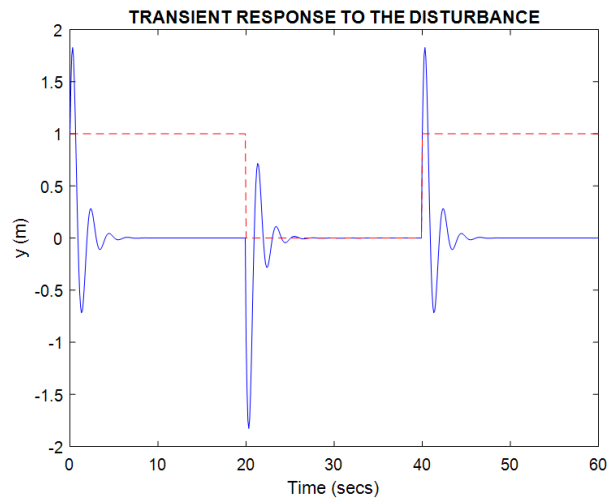


Figure 36: Transient response to disturbance input (K_{hin})

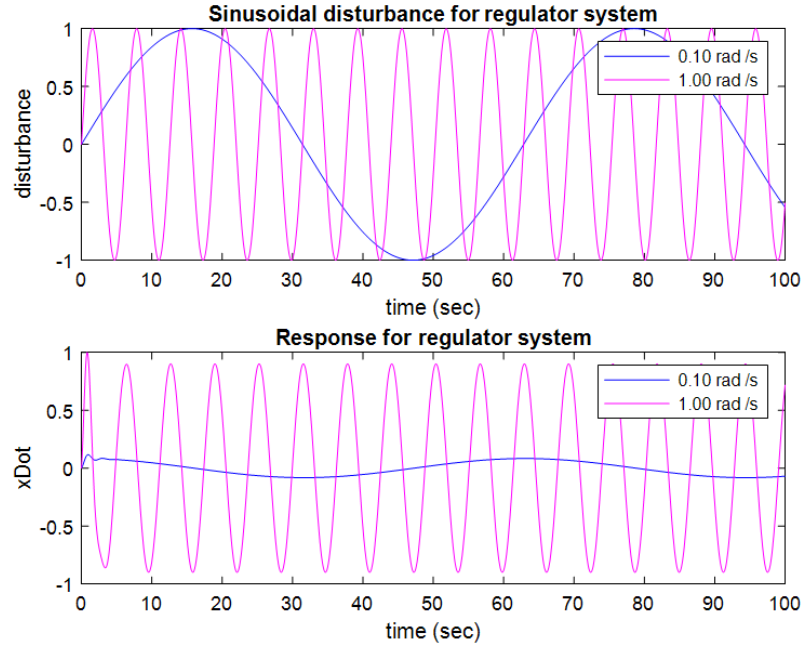


Figure 37: Sinusoidal disturbance for Regulator system for speed measurement

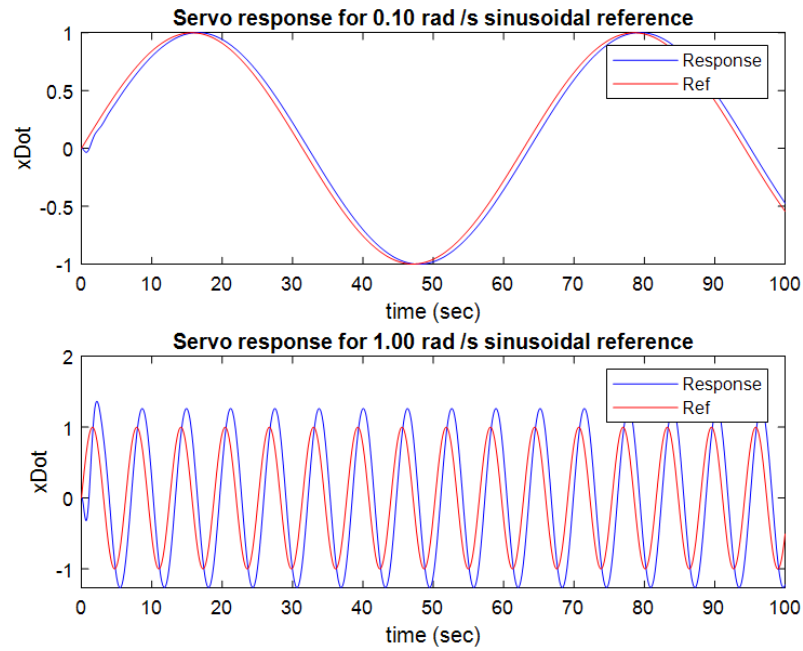


Figure 38: Servo response for different disturbances for speed measurement

Appendix

Part-A Regulator System

```
1  clc;clear all;close all;
2  %% Hinf controller design with unstructured uncertainty approach
3  % Part A) Regulation Problem with uncertainty -- Measurement is
    position
4  % Part B) Regulation Problem with uncertainty -- Measurement is
    speed
5  %% Part A) Regulation Problem with uncertainty -- Measurement is
    position
6  % Define Plant
7  num=1;
8  den=[1 1e-3 1e-6];
9  % Nominal plant (Transfer function with no uncertainty)
10 G0=tf(num,den);
11
12 s = tf('s');
13
14 % Parametric uncertainty
15 % Upper and lower bound for mass
16 m_upperBar = 1.2;
17 m_lowerBar = 0.8;
18
19 % Relative magnitude of the gain uncertainty
20 rm = (m_upperBar-m_lowerBar)/(m_upperBar+m_lowerBar);
21
22 % Any stable transfer function which at each frequency is less
23 % than or equal to one in magnitude.
24 delta = 1;
25
26 % Plant with multiplicative uncertainty
27 Gp = G0*(1+rm*delta);
28
29 % Add Actuator
30 % Delay-free nominal mode
31 U0 = 2.5/(2.5*s+1);
32 T = 4;
33 % Simple first-order weight
34 wI1 = (T*s + 0.2)/((T/2.5)*s+1);
35 deltaI = 1/(5*s+1)^3;
36 % deltaI = 0.1 / (s^2 + 0.1*s + 1);
37
38
39 % Multiply with correction factor to lift the gain
40 wI = wI1*tf([1 1.6 1],[1 1.4 1]);
41 Up = U0*(1+wI*deltaI);
42
43 % Plant with actuator
44 G = Up*Gp;
45
46 % Selecting weighting functions
47 % W1 is large frequency band where accurate tracking/disturbance
    rejection
```

```

48 % is required
49 wn=1e-3;
50 num_W1=2000^2;
51 den_W1=[1/wn^2 2*0.7/wn 1];
52 W1=tf(num_W1,den_W1);
53
54 % W2 is constant
55 num_W2=1e-3;
56 den_W2=1;
57 W2=tf(num_W2,den_W2);
58
59 % W3 is large frequency band where model is uncertain
60 num_W3=5e-8*[1/wn^2 2*0.7/wn 1];
61 den_W3=[1/200^2 2*0.7/200 1];
62 W3=tf(num_W3,den_W3);
63
64 % Calculation of augmented plant:
65 P = augw(G,W1,-W2,-W3);
66
67 % Calculation of optimal controller
68 nmeas=1; % number of inputs of the controller
69 ncon=1; % number of control inputs of the system P
70 [Kinf,Pcl,gamma_opt] = hinfsyn(P,nmeas,ncon);
71
72 L = G * Kinf;
73 % Sensitivity function
74 S = 1/(1 + L);
75 % Comp. Sensitivity function
76 T = L/(1 + L);
77
78 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Figures
79 % Bode Plot
80 figure
81 bode(1/W1)
82 hold on;
83 bode(1/W2)
84 hold on;
85 bode(1/W3)
86 hold on;
87
88 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
89 Gd = 1;
90 %
91 d = 1;
92 r = 0;
93
94 y = T*r + S*Gd*d;
95
96 % time variables
97 t = 0:0.1:100;
98 w = [0.1 0.5];
99
100 figure;
101 subplot(2,1,1)
102 % Plot Disturbance
103 for i = 1:length(w)

```



```

104     u_in = sin(w(i).*t);
105     [y_out,t_out,x] = lsim(y, u_in, t);
106     plot(t_out, u_in, 'LineWidth',1.25)
107     hold on;
108     legendStr(i) = sprintf(" %.2f rad/s", w(i));
109     xlabel('time(sec)');
110     ylabel('Position');
111     title('Sinusoidal disturbance for regulator system');
112 end
113 legend(legendStr);
114
115 hold on;
116 subplot(2,1,2)
117 % Plot Responses
118 for i = 1:length(w)
119     u_in = sin(w(i).*t);
120     [y_out,t_out,x] = lsim(y, u_in, t);
121     plot(t_out,y_out,'LineWidth',1.25);
122     legendStr(i) = sprintf(" %.2f rad/s", w(i));
123     xlabel('time(sec)');
124     ylabel('Position');
125     hold on;
126     title('Response for regulator system');
127 end
128 legend(legendStr);
129
130 % Uncertainty
131 % Ureal for mass
132 m_ureal = ureal('m_ureal',1,'Range',[0.8 1.2]);
133 % Ureal for actuator gain
134 U0_ureal = ureal('uo_ureal',1,'Range',[0.5 2]);
135 % Ureal for time constant
136 tau_ureal = ureal('tau_ureal',1,'Range',[0.5 2]);
137 % Time delay value
138 theta_ureal = 0.1;
139
140 % Plant
141 G_ureal = 1/(m_ureal*s^2);
142 % Actuator
143 U_ureal = U0_ureal * exp(-theta_ureal*s)/(tau_ureal*s + 1);
144
145 L_ureal = (U_ureal*G_ureal) * Kinf;
146 % Sensitivity function
147 S_ureal = 1 / (1 + L_ureal);
148 % Comp. Sensitivity function
149 T_ureal = L_ureal / (1 + L_ureal);
150
151 % plot the uncertain reponse of the controller
152 figure;
153 y_ureal = T_ureal * r + S_ureal * d;
154 step(y_ureal)
155 xlabel('Time');
156 ylabel('x');
157 title('Regulator step response for uncertain model');
158
159 % time variables

```

```

160 t = 0:0.1:100;
161 w = [0.1 1];
162
163 figure;
164 subplot(2,1,1)
165 % Plot Disturbance
166 for i = 1:length(w)
167     u_in = sin(w(i).*t);
168     [y_out_ureal,t_out,x] = lsim(y_ureal, u_in, t);
169     plot(t_out, u_in, 'LineWidth',1.25)
170     hold on;
171     legendStr(i) = sprintf(" %.2f rad/s", w(i));
172     xlabel('time(sec)');
173     ylabel('Position');
174     title('Sinusoidal disturbance for regulator system with
        uncertainty');
175 end
176 legend(legendStr);
177
178 hold on;
179 subplot(2,1,2)
180 % Plot Responses
181 for i = 1:length(w)
182     u_in = sin(w(i).*t);
183     [y_out_ureal,t_out,x] = lsim(y_ureal, u_in, t);
184     plot(t_out,y_out_ureal,'LineWidth',1.25);
185     legendStr(i) = sprintf(" %.2f rad/s", w(i));
186     xlabel('time(sec)');
187     ylabel('Position');
188     hold on;
189     title('Response for regulator system with uncertainty for
        position measurement');
190 end
191 legend(legendStr);
192
193 %% Plot mixed scenarios for position
194 r = 1;
195 d = 5;
196 y = T_ureal * r + S_ureal * d;
197 figure;
198 for i = 1:length(w)
199     uin = sin(w(i).*t);
200     subplot(2,1,i)
201     plot(t,uin,'--','LineWidth',1.25);
202     hold on;
203     [y_Out,t_Out,x] = lsim(y, uin, t);
204     plot(t_Out,y_Out,'LineWidth',1.25);
205     legend('Ref','Response')
206     xlabel('time (sec)');
207     ylabel('x');
208     title(sprintf('Mixed scenario with %.2f rad/s sinusoidal
        reference and disturbance',w(i)));
209 end
210
211 %% Part B) Regulation Problem with uncertainty -- Measurement is
        speed

```

```

212 % Define Plant
213 num = 1;
214 den = [1 1e-3];
215 % Nominal plant (Transfer function with no uncertainty)
216 G0 = tf(num,den);
217
218 s = tf('s');
219
220 % Parametric uncertainty
221 % Upper and lower bound for mass
222 m_upperBar = 1.2;
223 m_lowerBar = 0.8;
224
225 % Relative magnitude of the gain uncertainty
226 rm = (m_upperBar-m_lowerBar)/(m_upperBar+m_lowerBar);
227
228 % Any stable transfer function which at each frequency is less
229 % than or equal to one in magnitude.
230 delta = 1;
231
232 % Plant with multiplicative uncertainty
233 Gp = G0*(1+rm*delta);
234
235 % Add Actuator
236 % Delay-free nominal mode
237 U0 = 2.5/(2.5*s+1);
238 T = 4;
239 % Simple first-order weight
240 wI1 = (T*s + 0.2)/((T/2.5)*s+1);
241 deltaI = 1/(5*s+1)^3;
242 % deltaI = 0.1 / (s^2 + 0.1*s + 1);
243
244 % Multiply with correction factor to lift the gain
245 wI = wI1*tf([1 1.6 1],[1 1.4 1]);
246 Up = U0*(1+wI*deltaI);
247
248 % Plant with actuator
249 G = Up*Gp;
250
251 % Selecting weighting functions
252 % W1 is large frequency band where accurate tracking/disturbance
    rejection
253 % is required
254 wn = 1e-3;
255 num_W1 = 2000^2;
256 den_W1 = [1/wn^2 2*0.7/wn 1];
257 W1 = tf(num_W1,den_W1);
258
259 % W2 is constant
260 num_W2 = 1e-3;
261 den_W2 = 1;
262 W2 = tf(num_W2,den_W2);
263
264 % W3 is large frequency band where model is uncertain
265 num_W3 = 5e-8*[1/wn^2 2*0.7/wn 1];
266 den_W3 = [1/200^2 2*0.7/200 1];

```

```

267 W3 = tf(num_W3,den_W3);
268
269 % Calculation of augmented plant:
270 P = augw(G,W1,-W2,-W3);
271
272 % Calculation of optimal controller
273 nmeas = 1; % number of inputs of the controller
274 ncon = 1; % number of control inputs of the system P
275 [Kinf,Pcl,gamma_opt] = hinfsyn(P,nmeas,ncon);
276
277 L = G * Kinf;
278 % Sensitivity function
279 S = 1/(1 + L);
280 % Comp. Sensitivity function
281 T = L/(1 + L);
282
283 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Figures
284 % % Bode Plot
285 % figure
286 % bode(1/W1)
287 % hold on;
288 % bode(1/W2)
289 % hold on;
290 % bode(1/W3)
291 % hold on;
292
293 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
294 Gd = 1;
295 d = 1;
296 r = 0;
297
298 y = T*r + S*Gd*d;
299
300 % time variables
301 t = 0:0.1:100;
302 w = [0.1 0.5];
303
304 figure;
305 subplot(2,1,1)
306 % Plot Disturbance
307 for i = 1:length(w)
308     u_in = sin(w(i).*t);
309     [y_out,t_out,x] = lsim(y, u_in, t);
310     plot(t_out, u_in, 'LineWidth',1.25)
311     hold on;
312     legendStr(i) = sprintf(" %.2f rad/s", w(i));
313     xlabel('time(sec)');
314     ylabel('Speed');
315     title('Sinusoidal disturbance for regulator system');
316 end
317 legend(legendStr);
318
319 hold on;
320 subplot(2,1,2)
321 % Plot Responses
322 for i = 1:length(w)

```

```

323     u_in = sin(w(i).*t);
324     [y_out,t_out,x] = lsim(y, u_in, t);
325     plot(t_out,y_out,'LineWidth',1.25);
326     legendStr(i) = sprintf(" %.2f rad/s", w(i));
327     xlabel('time(sec)');
328     ylabel('Speed');
329     hold on;
330     title('Response for regulator system');
331 end
332 legend(legendStr);
333
334 % Uncertainty
335 % Ureal for mass
336 m_ureal = ureal('m_ureal',1,'Range',[0.8 1.2]);
337 % Ureal for actuator gain
338 U0_ureal = ureal('uo_ureal',1,'Range',[0.5 1.5]);
339 % Ureal for time constant
340 tau_ureal = ureal('tau_ureal',1,'Range',[0.5 2.5]);
341 % Time delay value
342 theta_ureal = 0.1;
343
344 % Plant
345 G_ureal = 1/(m_ureal*s);
346 % Actuator
347 U_ureal = U0_ureal * exp(-theta_ureal*s)/(tau_ureal*s + 1);
348
349 L_ureal = (U_ureal*G_ureal) * Kinf;
350 % Sensitivity function
351 S_ureal = 1 / (1 + L_ureal);
352 % Comp. Sensitivity function
353 T_ureal = L_ureal / (1 + L_ureal);
354
355 % plot the uncertain reponse of the controller
356 figure;
357 y_ureal = T_ureal * r + S_ureal * d;
358 step(y_ureal)
359 xlabel('Time');
360 ylabel('xDot');
361 title('Regulator step response for uncertain model');
362
363 % time variables
364 t = 0:0.1:100;
365 w = [0.1 1];
366
367 figure;
368 subplot(2,1,1)
369 % Plot Disturbance
370 for i = 1:length(w)
371     u_in = sin(w(i).*t);
372     [y_out_ureal,t_out,x] = lsim(y_ureal, u_in, t);
373     plot(t_out, u_in, 'LineWidth',1.25)
374     hold on;
375     legendStr(i) = sprintf(" %.2f rad/s", w(i));
376     xlabel('time(sec)');
377     ylabel('Speed');

```

```

378     title('Sinusoidal disturbance for regulator system with
          uncertainty');
379 end
380 legend(legendStr);
381
382 hold on;
383 subplot(2,1,2)
384 % Plot Responses
385 for i = 1:length(w)
386     u_in = sin(w(i).*t);
387     [y_out_ureal,t_out,x] = lsim(y_ureal, u_in, t);
388     plot(t_out,y_out_ureal,'LineWidth',1.25);
389     legendStr(i) = sprintf(" %.2f rad/s", w(i));
390     xlabel('time(sec)');
391     ylabel('Speed');
392     hold on;
393     title('Response for regulator system with uncertainty');
394 end
395 legend(legendStr);
396
397
398 %% Mixed scenarios
399 r = 1;
400 d = 5;
401 y = T_ureal * r + S_ureal * d;
402 figure;
403 for i = 1:length(w)
404     uin = sin(w(i).*t);
405     subplot(2,1,i)
406     plot(t,uin,'--','LineWidth',1.25);
407     hold on;
408     [y_Out,t_Out,x] = lsim(y, uin, t);
409     plot(t_Out,y_Out,'LineWidth',1.25);
410     legend('Ref','Response')
411     xlabel('Time (sec)');
412     ylabel('xDot');
413     title(sprintf('Mixed scenario with %.2f rad/s sinusoidal
          reference and disturbance',w(i)));
414 end

```

Part-A Servo System

```

1  clc;clear all;close all;
2  %% Hinf controller design with unstructured uncertainty approach
3  % Part A) Servo Problem with uncertainty -- Measurement is position
4  % Part B) Servo Problem with uncertainty -- Measurement is speed
5  %% Part A) Servo Problem with uncertainty -- Measurement is
   position
6  % Define Plant
7  num=1;
8  den=[1 1e-3 1e-6];
9  G0=tf(num,den);
10
11 s = tf('s');
12
13 % Parametric uncertainty

```

```

14 % Upper and lower bound for mass
15 m_upperBar = 1.2;
16 m_lowerBar = 0.8;
17
18 % Relative magnitude of the gain uncertainty
19 rm = (m_upperBar-m_lowerBar)/(m_upperBar+m_lowerBar);
20
21 % Any stable transfer function which at each frequency is less
22 % than or equal to one in magnitude.
23 delta = 1;
24 Gp = G0*(1+rm*delta);
25
26 % Actuator
27 U0 = 2.5/(2.5*s+1);
28 % Simple first-order weight
29 T = 4;
30 wI1 = (T*s + 0.2)/((T/2.5)*s+1);
31 deltaI = 1/(5*s+1)^3;
32 % deltaI = 0.1 / (s^2 + 0.1*s + 1);
33
34 % Multiply with correction factor to lift the gain
35 wI = wI1*tf([1 1.6 1],[1 1.4 1]);
36 Up = U0*(1+wI*deltaI);
37
38 % Plant with actuator
39 G = Up*Gp;
40
41 % Selecting weighting functions
42 % W1 is large frequency band where accurate tracking/disturbance
43 % rejection
44 % is required
45 wn = 1e-3;
46 num_W1 = 2000^2;
47 den_W1 = [1/wn^2 2*0.7/wn 1];
48 W1 = tf(num_W1,den_W1);
49
50 % W2 is constant
51 num_W2 = 1e-3;
52 den_W2 = 1;
53 W2 = tf(num_W2,den_W2);
54
55 % W3 is large frequency band where model is uncertain
56 num_W3 = 5e-8*[1/wn^2 2*0.7/wn 1];
57 den_W3 = [1/200^2 2*0.7/200 1];
58 W3 = tf(num_W3,den_W3);
59
60 % Calculation of augmented plant:
61 P = augw(G,W1,W2,W3);
62
63 % Calculation of optimal controller
64 nmeas = 1;
65 ncon = 1;
66 [Kinf,Pcl,gamma_opt] = hinfsyn(P,nmeas,ncon);
67
68 L = G*Kinf;
69 % Sensitivity function

```

```

69 S = 1/(1 + L);
70 % Comp. Sensitivity function
71 T = L/(1 + L);
72 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Figures
73 % Bode Plot
74 figure
75 bode(1/W1)
76 hold on;
77 bode(1/W2)
78 hold on;
79 bode(1/W3)
80 hold on;
81
82 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
83 Gd = 1;
84 % disturbance with no reference (Servo)
85 d = 0;
86 r = 1;
87
88 y = T*r + S*Gd*d;
89
90 % time variables
91 t = 0:0.1:100;
92 w = [0.1 1];
93
94 figure;
95 for i = 1:length(w)
96     u_in = sin(w(i).*t);
97     [y_out,t_out,x] = lsim(y, u_in, t);
98     subplot(2,1,i)
99     plot(t_out, u_in, 'LineWidth',1.25)
100    hold on;
101    plot(t_out,y_out,'LineWidth',1.25);
102    hold on;
103    legend('Reference','Response')
104    xlabel('time(sec)');
105    ylabel('Position');
106    title(sprintf('%.2f rad/s sinusoidal input for servo system',w(
        i)));
107 end
108
109 % Uncertainty
110 % Ureal for mass
111 m_ureal = ureal('m_ureal',1,'Range',[0.8 1.2]);
112 % Ureal for actuator gain
113 U0_ureal = ureal('uo_ureal',1,'Range',[0.5 1.5]);
114 % Ureal for time constant
115 tau_ureal = ureal('tau_ureal',1,'Range',[0.5 2.5]);
116 % Time delay value
117 theta_ureal = 0.1;
118
119 % Plant
120 G_ureal = 1/(m_ureal*s^2);
121 % Actuator
122 U_ureal = U0_ureal * exp(-theta_ureal*s)/(tau_ureal*s + 1);
123

```



```

124 L_ureal = (U_ureal*G_ureal) * Kinf;
125 % Sensitivity function
126 S_ureal = 1 / (1 + L_ureal);
127 % Comp. Sensitivity function
128 T_ureal = L_ureal / (1 + L_ureal);
129
130 % plot the uncertain reponse of the controller
131 figure;
132
133 y_ureal = T_ureal * r + S_ureal * d;
134 step(y_ureal)
135 xlabel('Time');
136 ylabel('x');
137 title('Servo step response for uncertain model');
138
139 % time variables
140 t = 0:0.1:100;
141 w = [0.1 1];
142
143 figure;
144 for i = 1:length(w)
145     u_in = sin(w(i).*t);
146     [y_out_ureal,t_out,x] = lsim(y_ureal, u_in, t);
147     subplot(2,1,i)
148     plot(t_out, u_in, 'LineWidth',1.25)
149     hold on;
150     plot(t_out,y_out_ureal,'LineWidth',1.25);
151     hold on;
152     legend('Reference','Response')
153     xlabel('time(sec)');
154     ylabel('Speed');
155     title(sprintf('%.2f rad/s sinusoidal input for servo system
with uncertainty',w(i)));
156
157 end
158
159 %% Part B) Servo Problem with uncertainty -- Measurement is speed
160 % Define Plant
161 num=1;
162 den=[1 1e-3];
163 G0=tf(num,den);
164
165 s = tf('s');
166
167 % Parametric uncertainty
168 % Upper and lower bound for mass
169 m_upperBar = 1.2;
170 m_lowerBar = 0.8;
171
172 % Relative magnitude of the gain uncertainty
173 rm = (m_upperBar-m_lowerBar)/(m_upperBar+m_lowerBar);
174
175 % Any stable transfer function which at each frequency is less
176 % than or equal to one in magnitude.
177 delta = 1;
178 Gp = G0*(1+rm*delta);

```

```

179
180 % Actuator
181 U0 = 2.5/(2.5*s+1);
182 % Simple first-order weight
183 T = 4;
184 wI1 = (T*s + 0.2)/((T/2.5)*s+1);
185 deltaI = 1/(5*s+1)^3;
186 % Multiply with correction factor to lift the gain
187 wI = wI1*tf([1 1.6 1],[1 1.4 1]);
188 Up = U0*(1+wI*deltaI);
189
190 % Plant with actuator
191 G = Up*Gp;
192
193 % Selecting weighting functions
194 % W1 is large frequency band where accurate tracking/disturbance
    rejection
195 % is required
196 wn = 1e-3;
197 num_W1 = 2000^2;
198 den_W1 = [1/wn^2 2*0.7/wn 1];
199 W1 = tf(num_W1,den_W1);
200
201 % W2 is constant
202 num_W2 = 1e-3;
203 den_W2 = 1;
204 W2 = tf(num_W2,den_W2);
205
206 % W3 is large frequency band where model is uncertain
207 num_W3 = 5e-8*[1/wn^2 2*0.7/wn 1];
208 den_W3 = [1/200^2 2*0.7/200 1];
209 W3 = tf(num_W3,den_W3);
210
211 % Calculation of augmented plant:
212 P = augw(G,W1,W2,W3);
213
214 % Calculation of optimal controller
215 nmeas=1;
216 ncon=1;
217 [Kinf,Pcl,gamma_opt] = hinfsyn(P,nmeas,ncon);
218
219 L = G*Kinf;
220 % Sensitivity function
221 S = 1/(1 + L);
222 % Comp. Sensitivity function
223 T = L/(1 + L);
224 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Figures
225 % Bode Plot
226 bode(1/W1)
227 hold on;
228 bode(1/W2)
229 hold on;
230 bode(1/W3)
231 hold on;
232
233 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```

234 Gd = 1;
235 % disturbance with no reference (Servo)
236 d = 0;
237 r = 1;
238
239 y = T*r + S*Gd*d;
240
241 % time variables
242 t = 0:0.1:100;
243 w = [0.1 1];
244
245 figure;
246 for i = 1:length(w)
247     u_in = sin(w(i).*t);
248     [y_out,t_out,x] = lsim(y, u_in, t);
249     subplot(2,1,i)
250     plot(t_out, u_in, 'LineWidth',1.25)
251     hold on;
252     plot(t_out,y_out,'LineWidth',1.25);
253     hold on;
254     legend('Reference','Response')
255     xlabel('time(sec)');
256     ylabel('Speed');
257     title(sprintf('%.2f rad/s sinusoidal input for servo system',w(
        i)));
258
259 end
260
261
262 % Uncertainty
263 % Ureal for mass
264 m_ureal = ureal('m_ureal',1,'Range',[0.8 1.2]);
265 % Ureal for actuator gain
266 U0_ureal = ureal('uo_ureal',1,'Range',[0.5 1.5]);
267 % Ureal for time constant
268 tau_ureal = ureal('tau_ureal',1,'Range',[0.5 2.5]);
269 % Time delay value
270 theta_ureal = 0.1;
271
272 % Plant
273 G_ureal = 1/(m_ureal*s);
274 % Actuator
275 U_ureal = U0_ureal * exp(-theta_ureal*s)/(tau_ureal*s + 1);
276
277 L_ureal = (U_ureal*G_ureal) * Kinf;
278 % Sensitivity function
279 S_ureal = 1 / (1 + L_ureal);
280 % Comp. Sensitivity function
281 T_ureal = L_ureal / (1 + L_ureal);
282
283 % plot the uncertain reponse of the controller
284 figure;
285
286 y_ureal = T_ureal * r + S_ureal * d;
287 step(y_ureal)
288 xlabel('Time');

```

```

289 ylabel('xDot');
290 title('Servo step response for uncertain model');
291
292 % time variables
293 t = 0:0.1:100;
294 w = [0.1 1];
295
296 figure;
297 for i = 1:length(w)
298     u_in = sin(w(i).*t);
299     [y_out_ureal,t_out,x] = lsim(y_ureal, u_in, t);
300     subplot(2,1,i)
301     plot(t_out, u_in, 'LineWidth',1.25)
302     hold on;
303     plot(t_out,y_out_ureal,'LineWidth',1.25);
304     hold on;
305     legend('Reference','Response')
306     xlabel('time(sec)');
307     ylabel('Speed');
308     title(sprintf('%.2f rad/s sinusoidal input for servo system
with Uncertainty',w(i)));
309
310 end

```

Part-B Code for Position Measurement

```

1  clc;clear; close all;
2  %% System Model
3  % Mass
4  m = 3;
5  % Time constant
6  tau = 1;
7  % Perturbation for mass
8  pm = 0.4;
9  % Perturbation for tau
10 ptau = 0.3;
11 %% State-space representation for G_mds
12 A = [ 0 1 0; 0 0 1/m ; 0 0 -1/tau];
13 B1 = [ 0 0; -pm 0 ;0 -ptau];
14 B2 = [ 0; 0; 1/tau];
15 C1 = [0 0 1/m; 0 0 -1\tau];
16 C2 = [ 1 0 0];
17 D11 = [-pm 0; 0 -ptau];
18 D12 = [0; 1\tau];
19 D21 = [0 0];
20 D22 = 0;
21
22 G = pck(A,[B1,B2],[C1;C2],[D11 D12;D21 D22]);
23
24 % This part is implemented to not see a error!!
25 % Unpack G and see the eignevalues
26 [a,b,c,d] = unpck(G);
27 eig(a)
28 % We do not want zero valued eigenvalue!!
29 % Zero eigenvalue leads to error
30 % Therefore, we add a very small number to change the zero

```

```

    eigenvalue
31 % situation
32 a(2,1)= 0.0001;
33 eig(a)
34 G = pck(a,b,c,d);
35
36 %% Frequency responses of the perturbed plants
37 omega = logspace(-1,1,100);
38 [delta1,delta2] = ndgrid([-1 0 1],[-1 0 1]);
39 for j = 1:9
40     delta = diag([delta1(j),delta2(j)]);
41     olp = starp(delta,G);
42     olp_ic = sel(olp,1,1);
43     olp_g = frsp(olp_ic,omega);
44     figure(1)
45     vplot('bode',olp_g,'c-')
46     subplot(2,1,1)
47     hold on
48     subplot(2,1,2)
49     hold on
50 end
51 subplot(2,1,1)
52 olp_ic = sel(G,3,3);
53 olp_g = frsp(olp_ic,omega);
54 vplot('bode',olp_g,'r--')
55 subplot(2,1,1)
56 title('BODE PLOTS OF PERTURBED PLANTS')
57 hold off
58 subplot(2,1,2)
59 hold off
60
61 %% Design Requirements of Closed-loop System
62 % Robust performance:
63 % Define the chosen weighting functions
64 nuWp = [1 1.8 10];
65 dnWp = [1 8 0.01];
66 gainWp = 0.95;
67 Wp = nd2sys(nuWp,dnWp,gainWp);
68 nuWu = 1;
69 dnWu = 1;
70 gainWu = 50^(-3);
71 Wu = nd2sys(nuWu,dnWu,gainWu);
72 % Calculate the inverse weighting function
73 omega = logspace(-4,4,100);
74 Wp_g = frsp(Wp,omega);
75 Wpi_g = minv(Wp_g);
76 figure
77 vplot('liv,lm',Wpi_g)
78 title('Inverse of Performance Weighting Function')
79 xlabel('Frequency (rad/sec)')
80 ylabel('Magnitude')
81
82 %% System Interconnections
83 systemnames = ' G Wp Wu ';
84 inputvar = '[ pert{2}; dist; control ]';
85 outputvar = '[ G(1:2); Wp; -Wu; -G(3)-dist ]';

```

```

86 input_to_G = '[ pert; control ]';
87 input_to_Wp = '[ G(3)+dist ]';
88 input_to_Wu = '[ control ]';
89 sysoutname = 'sys_ic';
90 cleanupsysic = 'no';
91 % create the structure of open-loop systems
92 sysic
93
94 % To analyse the open-loop system, the following commands can be
    used.
95 minfo(sys_ic)
96 spoles(sys_ic)
97 spoles(Wp)
98
99 % The model of the open-loop system with uncertainties is set
100 systemnames = ' G ';
101 inputvar = '[ pert{2}; ref; dist; control ]';
102 outputvar = '[ G(1:2); G(3)+dist; ref - G(3) - dist ]';
103 input_to_G = '[ pert; control ]';
104 sysoutname = 'sim_ic';
105 cleanupsysic = 'yes';
106 sysic
107
108 %% 8.5 Suboptimal H Controller Design
109 % number of measurements
110 nmeas = 1;
111 % number of controls
112 ncon = 1;
113 % lower bound of bisection
114 gmin = 1;
115 % upper bound of bisection
116 gmax = 10;
117 % absolute tolerance for the bisection method
118 tol = 0.001;
119 % open-loop interconnection is saved in the variable "hin_ic"
120 hin_ic = sel(sys_ic,3:5,3:4);
121 % K_hin: controller (matrix of type SYSTEM)
122 % clp: closed-loop system (matrix of type SYSTEM)
123 [K_hin,clp] = hinfsyn(hin_ic,nmeas,ncon,gmin,gmax,tol);
124
125 %% Analysis of Closed-loop System with Khin
126 % Singular values of the closed-loop system with Khin
127 minfo(K_hin)
128 spoles(K_hin)
129 omega = logspace(-2,6,100);
130 clp_g = frsp(clp,omega);
131 vplot('liv,lm',svsd(clp_g))
132 title('Singular Value Plot of clp')
133 xlabel('Frequency (rad/sec)')
134 ylabel('Magnitude')
135
136 % Sensitivity function with Khin plot
137 K = K_hin;
138 clp = starp(sim_ic,K);
139
140 % inverse performance weighting function

```

```

141 omega = logspace(-4,2,100);
142 Wp_g = frsp(Wp,omega);
143 Wpi_g = minv(Wp_g);
144
145 % sensitivity function
146 sen_loop = sel(clp,3,4);
147 sen_g = frsp(sen_loop,omega);
148 vplot('liv,lm',Wpi_g,'m--',vnorm(sen_g),'y-')
149 title('CLOSED-LOOP SENSITIVITY FUNCTION')
150 xlabel('Frequency (rad/sec)')
151 ylabel('Magnitude')
152
153 % Robust stability analysis of Khin
154 clp_ic = starp(sys_ic,K);
155 omega = logspace(-1,2,100);
156 clp_g = frsp(clp_ic,omega);
157 blkrsR = [-1 1;-1 1;-1 1];
158 rob_stab = sel(clp_g,[1:3],[1:3]);
159 pdim = ynum(rob_stab);
160 fixl = [eye(pdim); 0.1*eye(pdim)]; % 1% Complex
161 fixr = fixl';
162 blkrs = [blkrsR; abs(blkrsR)];
163 clp_mix = mmult(fixl,rob_stab,fixr);
164 [rbnds,rowd,sens,rowp,rowg] = mu(clp_mix,blkrs);
165 disp(' ')
166 disp(['mu-robust stability: ' ...
167 num2str(pkvnorm(sel(rbnds,1,1))))])
168 disp(' ')
169 vplot('liv,lm',sel(rbnds,1,1),'y--',sel(rbnds,1,2),'m-', ...
170 vnorm(rob_stab),'c-.')
171 title('ROBUST STABILITY')
172 xlabel('Frequency (rad/s)')
173 ylabel('mu')
174
175 % Nominal and robust performance of Khin
176 clp_ic = starp(sys_ic,K);
177 omega = logspace(-1,2,100);
178 clp_g = frsp(clp_ic,omega);
179 blkrsR = [-1 1;-1 1];
180
181 % Nominal performance
182 nom_perf = sel(clp_g,3,3);
183
184 % Robust performance
185 rob_perf = clp_g;
186 blkrp = [blkrsR;[1 2]];
187 bndsrp = mu(rob_perf,blkrp);
188 vplot('liv,lm',vnorm(nom_perf),'y-',sel(bndsrp,1,1),'m--',...
189 sel(bndsrp,1,2),'c--')
190 tmp1 = 'NOMINAL PERFORMANCE (solid) and';
191 tmp2 = ' ROBUST PERFORMANCE (dashed)';
192 title([tmp1 tmp2])
193 xlabel('Frequency (rad/s)')
194 disp(' ')
195 disp(['mu-robust performance: ' ...
196 num2str(pkvnorm(sel(bndsrp,1,1))))])

```

```

197 disp(' ')
198
199 % Transient response to reference/disturbance input
200 clp = starp(sim_ic,K_hin);
201 timedata = [0 20 40];
202 stepdata = [1 0 1];
203 dist = 0;
204 ref = step_tr(timedata,stepdata,0.1,60);
205 u = abv(0,0,ref,dist);
206 y = trsp(clp,u,60,0.1);
207 figure
208 vplot(sel(y,3,1),'y-',ref,'r--')
209 title('CLOSED-LOOP TRANSIENT RESPONSE')
210 xlabel('Time (secs)')
211 ylabel('y (m)')
212
213 % Response to the disturbance
214 timedata = [0 20 40];
215 stepdata = [1 0 1];
216 dist = step_tr(timedata,stepdata,0.1,60);
217 ref = 0;
218 u = abv(0,0,ref,dist);
219 y = trsp(clp,u,60,0.1);
220 figure
221 vplot(sel(y,3,1),'y-',dist,'r--')
222 title('TRANSIENT RESPONSE TO THE DISTURBANCE')
223 xlabel('Time (secs)')
224 ylabel('y (m)')
225
226 %% Servo Problem measurement speed
227 clp = starp(sim_ic,K_hin);
228 w = [0.1, 1];
229 dist = 0;
230 figure;
231 for i = 1:length(w)
232     freq = w(i);
233     ref = sin_tr (freq ,1 ,0.1 ,100) ;
234     u = abv(0,0 ,ref , dist );
235     y = trsp(clp ,u ,100 ,0.1) ;
236     subplot (2,1,i);
237     vplot ( sel(y,3,1),'b',ref , 'r')
238     xlabel ('time (sec)');
239     ylabel ('x');
240     legend ('Response','Ref ')
241     title (sprintf('Servo response for %.2f rad /s sinusoidal
reference',w(i)));
242 end
243
244 %% Regulator Problem
245 clp = starp(sim_ic ,K_hin);
246 w = [0.1, 1];
247 ref = 0;
248 figure ;
249 for i = 1: length(w)
250     freq = w(i);
251     dist = sin_tr(freq ,1 ,0.1 ,100) ;

```



```

252     u = abv (0,0 ,ref , dist );
253     y = trsp (clp ,u ,100 ,0.1) ;
254
255     % Sinusoidal disturbance for regulator system
256     subplot (2 ,1 ,1);
257     vplot (dist);
258     hold on;
259     xlabel ('time (sec)');
260     ylabel ('disturbance');
261     title ('Sinusoidal disturbance for regulator system');
262
263     % Response for regulator system
264     subplot (2 ,1 ,2)
265     vplot (sel(y ,3 ,1))
266     hold on;
267     legendStr (i) = sprintf (" %.2f rad /s", w(i));
268     xlabel ('time (sec)');
269     ylabel ('x');
270     title ('Response for regulator system');
271 end
272 subplot (2 ,1 ,1)
273 legend (legendStr);
274 subplot (2 ,1 ,2)
275 legend (legendStr);

```

Part-B Code for Speed Measurement

```

1  clc;clear; close all;
2  %% System Model
3  % Mass
4  m = 3;
5  % Time constant
6  tau = 1;
7  % Perturbation for mass
8  pm = 0.4;
9  % Perturbation for tau
10 ptau = 0.3;
11 %% State-space representation for G_mds
12 % System matrices
13 A = [0  1/m; 0 -1/tau];
14 B1 = [-pm 0 ;0 -ptau];
15 B2 = [0; 1/tau];
16 C1 = [0 1/m;0 -1\tau];
17 C2 = [1 0];
18 D11 = [-pm 0; 0 -ptau];
19 D12 = [0; 1\tau];
20 D21 = [0 0];
21 D22 = 0;
22
23 G = pck(A,[B1,B2],[C1;C2],[D11 D12;D21 D22]);
24
25 % This part is implemented to not see a error!!
26 % Unpack G and see the eignevalues
27 [a,b,c,d] = unpck(G);
28 eig(a)
29 % We do not want zero valued eigenvalue!!

```

```

30 % Zero eigenvalue leads to error
31 % Therefore, we add a very small number to change the zero
    eigenvalue
32 % situation
33 a(2,1)= 0.0001;
34 eig(a)
35 G = pck(a,b,c,d);
36
37 %% Frequency responses of the perturbed plants
38 omega = logspace(-1,1,100);
39 [delta1,delta2] = ndgrid([-1 0 1],[-1 0 1]);
40 for j = 1:9
41     delta = diag([delta1(j),delta2(j)]);
42     olp = starp(delta,G);
43     olp_ic = sel(olp,1,1);
44     olp_g = frsp(olp_ic,omega);
45     figure(1)
46     vplot('bode',olp_g,'c-')
47     subplot(2,1,1)
48     hold on
49     subplot(2,1,2)
50     hold on
51 end
52 subplot(2,1,1)
53 olp_ic = sel(G,3,3);
54 olp_g = frsp(olp_ic,omega);
55 vplot('bode',olp_g,'r--')
56 subplot(2,1,1)
57 title('BODE PLOTS OF PERTURBED PLANTS')
58 hold off
59 subplot(2,1,2)
60 hold off
61
62 %% Design Requirements of Closed-loop System
63 % Robust performance:
64 % Define the chosen weighting functions
65 nuWp = [1 1.8 10];
66 dnWp = [1 8 0.01];
67 gainWp = 0.95;
68 Wp = nd2sys(nuWp,dnWp,gainWp);
69 nuWu = 1;
70 dnWu = 1;
71 gainWu = 50^(-3);
72 Wu = nd2sys(nuWu,dnWu,gainWu);
73 % Calculate the inverse weighting function
74 omega = logspace(-4,4,100);
75 Wp_g = frsp(Wp,omega);
76 Wpi_g = minv(Wp_g);
77 figure
78 vplot('liv,lm',Wpi_g)
79 title('Inverse of Performance Weighting Function')
80 xlabel('Frequency (rad/sec)')
81 ylabel('Magnitude')
82
83 %% System Interconnections
84 systemnames = ' G Wp Wu ';

```

```

85 inputvar = '[ pert{2}; dist; control ]';
86 outputvar = '[ G(1:2); Wp; -Wu; -G(3)-dist ]';
87 input_to_G = '[ pert; control ]';
88 input_to_Wp = '[ G(3)+dist ]';
89 input_to_Wu = '[ control ]';
90 sysoutname = 'sys_ic';
91 cleanupsysic = 'no';
92 % create the structure of open-loop systems
93 sysic
94
95 % To analyse the open-loop system, the following commands can be
    used.
96 minfo(sys_ic)
97 spoles(sys_ic)
98 spoles(Wp)
99
100 % The model of the open-loop system with uncertainties is set
101 systemnames = ' G ';
102 inputvar = '[ pert{2}; ref; dist; control ]';
103 outputvar = '[ G(1:2); G(3)+dist; ref - G(3) - dist ]';
104 input_to_G = '[ pert; control ]';
105 sysoutname = 'sim_ic';
106 cleanupsysic = 'yes';
107 sysic
108
109 %% 8.5 Suboptimal H Controller Design
110 % number of measurements
111 nmeas = 1;
112 % number of controls
113 ncon = 1;
114 % lower bound of bisection
115 gmin = 1;
116 % upper bound of bisection
117 gmax = 10;
118 % absolute tolerance for the bisection method
119 tol = 0.001;
120 % open-loop interconnection is saved in the variable "hin_ic"
121 hin_ic = sel(sys_ic,3:5,3:4);
122 % K_hin: controller (matrix of type SYSTEM)
123 % clp: closed-loop system (matrix of type SYSTEM)
124 [K_hin,clp] = hinfsyn(hin_ic,nmeas,ncon,gmin,gmax,tol);
125
126 %% Analysis of Closed-loop System with Khin
127 % Singular values of the closed-loop system with Khin
128 minfo(K_hin)
129 spoles(K_hin)
130 omega = logspace(-2,6,100);
131 clp_g = frsp(clp,omega);
132 vplot('liv,lm',vsvd(clp_g))
133 title('Singular Value Plot of clp')
134 xlabel('Frequency (rad/sec)')
135 ylabel('Magnitude')
136
137 % Sensitivity function with Khin plot
138 K = K_hin;
139 clp = starp(sim_ic,K);

```

```

140
141 % inverse performance weighting function
142 omega = logspace(-4,2,100);
143 Wp_g = frsp(Wp,omega);
144 Wpi_g = minv(Wp_g);
145
146 % sensitivity function
147 sen_loop = sel(clp,3,4);
148 sen_g = frsp(sen_loop,omega);
149 vplot('liv,lm',Wpi_g,'m--',vnorm(sen_g),'y-')
150 title('CLOSED-LOOP SENSITIVITY FUNCTION')
151 xlabel('Frequency (rad/sec)')
152 ylabel('Magnitude')
153
154 % Robust stability analysis of Khin
155 clp_ic = starp(sys_ic,K);
156 omega = logspace(-1,2,100);
157 clp_g = frsp(clp_ic,omega);
158 blkrsR = [-1 1;-1 1;-1 1];
159 rob_stab = sel(clp_g,[1:3],[1:3]);
160 pdim = ynum(rob_stab);
161 fixl = [eye(pdim); 0.1*eye(pdim)]; % 1% Complex
162 fixr = fixl';
163 blkrs = [blkrsR; abs(blkrsR)];
164 clp_mix = mmult(fixl,rob_stab,fixr);
165 [rbnds,rowd,sens,rowp,rowg] = mu(clp_mix,blkrs);
166 disp(' ')
167 disp(['mu-robust stability: ' ...
168 num2str(pkvnorm(sel(rbnds,1,1))))]
169 disp(' ')
170 vplot('liv,lm',sel(rbnds,1,1),'y--',sel(rbnds,1,2),'m-', ...
171 vnorm(rob_stab),'c-.')
172 title('ROBUST STABILITY')
173 xlabel('Frequency (rad/s)')
174 ylabel('mu')
175
176 % Nominal and robust performance of Khin
177 clp_ic = starp(sys_ic,K);
178 omega = logspace(-1,2,100);
179 clp_g = frsp(clp_ic,omega);
180 blkrsR = [-1 1;-1 1];
181
182 % Nominal performance
183 nom_perf = sel(clp_g,3,3);
184
185 % Robust performance
186 rob_perf = clp_g;
187 blkrp = [blkrsR;[1 2]];
188 bndsrp = mu(rob_perf,blkrp);
189 vplot('liv,lm',vnorm(nom_perf),'y-',sel(bndsrp,1,1),'m--',...
190 sel(bndsrp,1,2),'c--')
191 tmp1 = 'NOMINAL PERFORMANCE (solid) and';
192 tmp2 = ' ROBUST PERFORMANCE (dashed)';
193 title([tmp1 tmp2])
194 xlabel('Frequency (rad/s)')
195 disp(' ')

```

```

196 disp(['mu-robust performance: ' ...
197 num2str(pkvnorm(sel(bndsrp,1,1))))]
198 disp(' ')
199
200 % Transient response to reference/disturbance input
201 clp = starp(sim_ic,K_hin);
202 timedata = [0 20 40];
203 stepdata = [1 0 1];
204 dist = 0;
205 ref = step_tr(timedata,stepdata,0.1,60);
206 u = abv(0,0,ref,dist);
207 y = trsp(clp,u,60,0.1);
208 figure
209 vplot(sel(y,3,1),'y-',ref,'r--')
210 title('CLOSED-LOOP TRANSIENT RESPONSE')
211 xlabel('Time (secs)')
212 ylabel('y (m)')
213
214 % Response to the disturbance
215 timedata = [0 20 40];
216 stepdata = [1 0 1];
217 dist = step_tr(timedata,stepdata,0.1,60);
218 ref = 0;
219 u = abv(0,0,ref,dist);
220 y = trsp(clp,u,60,0.1);
221 figure
222 vplot(sel(y,3,1),'y-',dist,'r--')
223 title('TRANSIENT RESPONSE TO THE DISTURBANCE')
224 xlabel('Time (secs)')
225 ylabel('y (m)')
226
227 %% Servo Problem measurement speed
228 clp = starp(sim_ic,K_hin);
229 w = [0.1, 1];
230 dist = 0;
231 figure;
232 for i = 1:length(w)
233     freq = w(i);
234     ref = sin_tr (freq ,1 ,0.1 ,100) ;
235     u = abv(0,0 ,ref , dist );
236     y = trsp(clp ,u ,100 ,0.1) ;
237     subplot (2,1,i);
238     vplot (sel(y,3,1),'b',ref , 'r')
239     xlabel ('time (sec)');
240     ylabel ('xDot');
241     legend ('Response','Ref ')
242     title (sprintf('Servo response for %.2f rad /s sinusoidal
reference',w(i)));
243 end
244
245 %% Regulator Problem
246 clp = starp(sim_ic ,K_hin);
247 w = [0.1, 1];
248 ref = 0;
249 figure ;
250 for i = 1: length(w)

```

```

251     freq = w(i);
252     dist = sin_tr(freq ,1 ,0.1 ,100) ;
253     u = abv (0,0 ,ref , dist );
254     y = trsp (clp ,u ,100 ,0.1) ;
255
256     % Sinusoidal disturbance for regulator system
257     subplot (2 ,1 ,1);
258     vplot (dist);
259     hold on;
260     xlabel ('time (sec)');
261     ylabel ('disturbance');
262     title ('Sinusoidal disturbance for regulator system');
263
264     % Response for regulator system
265     subplot (2 ,1 ,2)
266     vplot (sel(y ,3 ,1))
267     hold on;
268     legendStr (i) = sprintf (" %.2f rad /s", w(i));
269     xlabel ('time (sec)');
270     ylabel ('xDot');
271     title ('Response for regulator system');
272 end
273 subplot (2 ,1 ,1)
274 legend (legendStr);
275 subplot (2 ,1 ,2)
276 legend (legendStr);

```